

AD A108976

LEVEL

10

AFOSR-TR- 81 -0845
A Hierarchical Pattern Extraction System

For
Hexagonally Sampled Images

Laurie D. Gibson
Charles Lenzmeier

Interactive Systems Corporation

October 1981

Final Report for Period 1 March 1981 - 1 September 1981

Prepared for:

THE AIR FORCE OFFICE OF SCIENTIFIC RESEARCH
Building 410
Bolling Air Force Base, D.C. 20332

AIR FORCE OFFICE OF SCIENTIFIC RESEARCH (AFOSR)
NOTICE OF TECHNICAL INFORMATION
This technical report is approved for
Distribution unlimited.
MATTHEW J. KEMPER
Chief, Technical Information Division

Approved for public release;
distribution unlimited.

81 12 29 032

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER AFOSR-TR- 81 -0845	2. GOVT ACCESSION NO. AD-A108 976	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) A Hierarchical Pattern Extraction System for Hexagonally Sampled Images		5. TYPE OF REPORT & PERIOD COVERED Final - 1 Mar 81 to 1 Sep 81
7. AUTHOR(s) Laurie D. Gibson Charles T. Lenzmeier		6. PERFORMING ORG. REPORT NUMBER
8. CONTRACT OR GRANT NUMBER(s) F49620-81-C-0039		9. PERFORMING ORGANIZATION NAME AND ADDRESS Interactive Systems Corporation 5500 South Sycamore Street Littleton, Colorado 80120
10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS 61102F; 2304/A2		11. CONTROLLING OFFICE NAME AND ADDRESS Directorate of Mathematical & Information Sciences Air Force Office of Scientific Research Building 410, Bolling AFB, DC 20332 Code: FQ8671
12. REPORT DATE October 1981		13. NUMBER OF PAGES 82
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office)		15. SECURITY CLASS. (of this report) Unclassified
15a. DECLASSIFICATION DOWNGRADING SCHEDULE		
16. DISTRIBUTION STATEMENT (of this Report) Approved for public release; distribution unlimited.		
17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)		
18. SUPPLEMENTARY NOTES		
19. KEY WORDS (Continue on reverse side if necessary and identify by block number) Image Processing; Pattern Recognition; Raster to Vector Conversion; Hierarchical Data Base Structures; Hexagonally Sampled Images		
20. ABSTRACT (Continue on reverse side if necessary and identify by block number) The objective of this research was to study the application of Generalized Balanced Ternary (GBT), a hierarchical coordinate system, and its related data structures to the problem of extracting line and area data (vectors) from rasterized images. GBT allows the pixels in an image to be aggregated over regions of different sizes. In the raster to vector process certain descriptors are calculated for these aggregates. They describe the general- OVER		

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

40

395387

LB

20. → raster pattern within the aggregate region. Algorithms were developed to vectorize by examining the descriptors rather than the pixel data. The algorithms were implemented in software and tested on sample data. The results indicate this approach is promising and suggest ways to improve the algorithms. ↗

TABLE OF CONTENTS

SECTION NUMBER	DESCRIPTION	PAGE
1.0	Introduction.....	1
2.0	Technical Problem.....	2
2.1	Existing Raster to Vector Conversion Algorithms.....	2
3.0	Scope.....	3
4.0	Theory.....	5
4.1	Raster to Vector Algorithms.....	5
4.1.1	Aggregate Descriptors.....	5
4.1.2	The Line-Following Algorithm.....	14
4.1.3	The Boundary-Following Algorithm.....	26
4.1.4	The Smoothing Algorithm.....	26
4.2	Generalized Balanced Ternary.....	29
4.2.1	The Structure and Addresses.....	29
4.2.2	The Arithmetic.....	36
4.2.3	The GBT Data Base Structure.....	42
4.2.4	Data Types and Representations.....	47
5.0	Implementation.....	49
5.	Test Data Sets.....	51
5.2	Conclusions.....	77
6.0	Future Research Directions.....	79
6.1	Prototype Automated Graphics Entry System.....	80
	References.....	82

LIST OF TABLES

1	128 Pattern Types.....	9 10 11 12 13
2	GBT Addition.....	38
3	GBT Multiplication.....	40

Accession for	<input checked="" type="checkbox"/> DTIC <input type="checkbox"/> DTIC <input type="checkbox"/> DTIC <input type="checkbox"/> DTIC <input type="checkbox"/> DTIC	By	Division/ Division Division Division Division
Available for Codes 1 and 2 3 and 4 5 and 6 7 and 8 9 and 10		<div style="font-size: 2em; font-weight: bold;">A</div>	

TABLE OF CONTENTS

LIST OF FIGURES

NUMBER	DESCRIPTION	PAGE
1	The Level Three Aggregate A has a Pathological Pattern Type.....	8
2	A Raster Scan of the Letters "OK".....	15
3	Level Two Weighted Centroids for "OK".....	16
4	Level Three Weighted Centroids for "OK".....	17
5	Level Two Gradients for "OK".....	18
6	Level Two Orientations for "OK".....	19
7	Level Three Orientations for "OK".....	20
8	Line-Following Pattern Types for Level Two Aggregates.....	21
9	Line-Following Pattern Types for Level Three Aggregates.....	22
10	Edge cells are Paired by Their Orientations.....	23
11	For this Chain of Four Cells a Weighted Centroid is Computed.....	24
12	A Transit Cell (center) and its Centroid.....	25
13	A is a Pathological Aggregate.....	27
14	The Centroid, Anti-centroid, and Boundary Centroid of an Aggregate	28
15	Smoothing the line ABCD.....	30
16	The Hexagonal Covering of the Plane.....	32
17	The Aggregate Structure.....	33
	17A First Level Aggregate.....	33
	17B Second Level Aggregate.....	33
	17C Third Level Aggregate.....	33
18	The Location of the Hexagon whose Address is 536.....	34
19	Two Examples of the Use of the Trailing Sevens Notation to Address Higher Aggregates.....	35
20	Key to Planar Addition Table.....	37
21	The GBT Sum of 153 and 45.....	39
22	The GBT Product: 254 x 62.....	41
23	A List of Addresses and the Corresponding Tree Structure.....	43
24	The Location of Cells on the Address List.....	44
25	An Abbreviated Tree Corresponding to the Same Address List as Figure 23.....	46
26	Data Types and their Associated GBT File Representations.....	48
27	The Experimental Raster to Vector Conversion System.....	50
28	A Portion of the USGS 7.5 Minute Contour Overlay used to Test the RVC System (DATA SET #1).....	52
29	A Portion of the Overlay of Streets and Buildings used in the System Test (DATA SET #2).....	53
30	A Portion of a Circuit Diagram for a Submarine (DATA SET #3).....	54
31	A Portion of a City Planning Map (DATA SET #4).....	55
32	Vectorization of a 25 Micron Scan (DATA SET #1).....	56
33	Close-up of Figure 32.....	57
34	Close-up of Figure 32.....	58

TABLE OF CONTENTS

LIST OF FIGURES (Cont.)

NUMBER	DESCRIPTION	PAGE
35	Close-up of Figure 32.....	59
36	Vectorization of a 25 Micron Scan (DATA SET #2).....	60
37	Vectorization of a 25 Micron Scan (DATA SET #2).....	61
38	Close-up of Figure 36.....	62
39	Close-up of Figure 37, unsmoothed.....	63
40	Same Area as in Figure 39, After Smoothing.....	64
41	Vectorization of a 100 Micron Scan (DATA SET #3).....	65
42	Close-up of Figure 41.....	66
43	Close-up of Figure 41.....	67
44	Vectorization of a 50 Micron Scan (DATA SET #4).....	68
45	Vectorization of a 50 Micron Scan (DATA SET #4).....	69
46	Vectorization of a 50 Micron Scan (DATA SET #4).....	70
47	Close-up of Figure 44.....	71
48	Close-up of Figure 45.....	72
49	Close-up of Figure 46.....	73
50	Ink Drawing Scanned for Boundary-Following.....	74
51	Vectorization of a 25 Micron Scan.....	75
52	Close-up of the Vectorization of Figure 50.....	76
53	Aggregate A for which $M_1(A)=0$	79
54	Aggregate A for which $M_1(A)=1$	79

1.0 Introduction

The objective of the research performed under this contract was to study the application of Generalized Balanced Ternary, a hierarchical coordinate system, and its related data structures to the problem of extracting line and area data from rasterized images. Algorithms were developed, implemented in software, tested, and refined. This was done using high resolution scans of maps and engineering drawings. The results indicate that this approach to building an automated system for the entry of graphical data is very promising.

Section 2 of this report discusses the general context of the research. In Section 3, the research plan is presented. Section 4 contains a description of Generalized Balanced Ternary and of the algorithms developed during this project. The software implementation is dealt with in Section 5, along with the final results on the experimental system. Section 6 addresses further theoretical considerations and the development of a prototype system.

2.0 Technical Problem

With the increasing use of computer graphics and of digital computers in processing spatial data, the problem of data capture has become more apparent. A large data base of graphical data, for example, engineering drawings, can be easily maintained in digital form. But the task of inputting an existing paper data base can be monumental. The usual method consists of manual digitization, tracing lines by hand. There are systems that attempt to automate this process to some extent. Using a scanner, either the lines in the image are followed or the image is captured in pixel form with a raster scan. In the second case, this raster image is manipulated (usually in software) by algorithms which extract lines or boundaries as sequences of vectors (raster to vector conversion). So far neither of these approaches has proven adequate to handle a variety of graphics data in a way that is quick, accurate, and inexpensive.

The purpose of the research described here is to try a new approach to the raster to vector conversion algorithms. It is hoped that this approach may be the basis for major improvement in the process of automated entry of graphics data.

2.1 Existing Raster to Vector Conversion Algorithms

Included here is a brief discussion of existing RVC algorithms and how the process has been structured by others. For a specific and detailed summary of RVC efforts see (1).

Consider an image as a binary pixel array of some unit resolution from which lines and the boundaries of regions are to be extracted. The first step in this process is to reduce all lines to a thickness of one resolution unit. This is done by making a series of passes through the array. On each pass, each pixel that is set to on is examined and it is decided whether or not to turn it off. How this is decided varies with the algorithm. As an example, the eight neighbors of a pixel may be examined. Certain configurations (those which suggest the pixel lies on the edge of a line) result in the pixel being set to off. Ideally, several passes will result in lines that are one pixel wide without distorting line junctions. The next step is to generate vector lists by moving from set pixel to set pixel. Line branchings are recognized and recorded at this point, permitting the vectors, finally, to be structured into a polygonal representation of map lines and regions. At some point the vectors should be cleaned up: spurs removed, breaks closed, and data volume reduced. Limiting the approach is the sheer number of pixels in a high resolution scan (400 million for an average sized map scanned at 25 microns). Any operation on the pixels must be fairly simple and fast. Because so much of the logic is on the pixel level, many of these algorithms are quite sensitive to variation in line widths and to the overall quality of the lines. When incorporated in a system, this translates to a great deal of time and effort being required to preedit the input data.

3.0 Scope

The research performed under this contract was done from 1 March 1981 to 1 September 1981 by Dr. Dean Lucas, Dr. Laurie Gibson, and Mr. Charles Lenzmeier. This effort built on previous work establishing the theoretic basis for Generalized Balanced Ternary and developing the algorithms which allow it to be applied to spatial data management systems. The research objectives were as follows:

- 1) Develop the theory and algorithms for raster to vector conversion using Generalized Balanced Ternary (GBT).
- 2) Design and build an experimental software package including:
 - a) a program to populate a hierarchical data base.
 - b) a program to produce vectors along line centers for line data and vectors along boundaries for area data.
 - c) a program to reduce vector volume by smoothing and to produce X-Y plot files.
- 3) Test the software on rasterized map data.

The experimental software was produced to interface with ISC's existing data management system and interactive graphics software. All software was written in Fortran and run on a VAX 11/780 operating under VMS. The graphics equipment used was a Megatek System 7210 monitor, a Terak black & white raster display, and a Versatec printer/plotter. The research proceeded in these phases:

- 1) Define the primitives to be stored in the data base and develop the algorithms required to compute the primitives from raster data.
- 2) Design a centerlining algorithm.
- 3) Build as a research tool the basic data management and graphics display system (this system was put together from existing ISC software in about a week).
- 4) Implement the algorithms developed under phases (1) and (2) and test them using analytically generated raster scans.
- 5) Refine the algorithms and software until the results on the simulated scans are satisfactory.
- 6) Test the software on a rasterized version of a dense contour map.
- 7) Refine algorithms and software until satisfactory results are obtained on this data set.
- 8) Begin working with scanned data from various kinds of scanners. Work with several types of map data.
- 9) Design the smoothing algorithm. Implement and test it.

- 10) Design the boundary-following algorithm, implement and test it (first on simulated scans and then on actual data).
- 11) Refine the boundary-following algorithm and software until satisfactory results are obtained.
- 12) Redesign the data base population program to optimize for speed.

4.0 Theory

Underlying this approach to raster to vector conversion is Generalized Balanced Ternary, a system for addressing regions in space. GBT is a method of representing a two dimensional surface that enables a computer to work easily with data distributed on that surface. When a person looks at data in two dimensions, a map sheet or a photographic image, for example, he quickly sees the general content of that map or photograph, and can examine certain aspects of the content in detail if he chooses to focus his attention. Computers which use conventional representations for planar data have difficulty performing this simple chore. Such systems permit easy examination of the smallest components of the data, be they pixels, bits, bytes, or vectors, but have no efficient mechanisms for examining the data in aggregate form. As the saying goes, they cannot see the forest for the trees.

GBT begins with a hexagonal array in the plane and assigns to each array point an address which is a finite integer string. This addressing scheme possesses its own arithmetic system and generates a very efficient tree data structure.

GBT has many advantages in the raster-to-vector conversion application. For one, the pixels in a hexagonal array have uniform adjacency. This avoids the problems of pixels which can be neighbors in more than one way as, for example, in a rectangular array. The ability to aggregate pixels is of great importance to the algorithms developed here. Instead of pixel-by-pixel processing, the image can be considered over various sized aggregate areas. This allows the extraction of general patterns and reduces the volume of basic elements to be processed. More complex algorithms can be run on relatively few aggregates. GBT supports these algorithms with a rich underlying mathematical structure. Functions which describe aggregate patterns can be defined with this arithmetic. The algorithms also use it to move from aggregate to aggregate in the array.

An introduction to GBT is included at the end of this chapter. It is intended for those unfamiliar with the system and should be read first.

4.1 Raster to Vector Algorithms

The raster to vector conversion process has two stages. First, the pixel data is generalized by computing the aggregate primitives or descriptors which describe the data in terms of meaningful pattern representations. The second stage consists of using these representations to extract line-following and edge-following vectors. We define the aggregate descriptors as follows.

4.1.1 Aggregate Descriptors

Suppose an image or picture is sampled in a GBT grid (hexagonally) generating a picture function F on the set of GBT addresses. Suppose A is a GBT aggregate. Define:

- 1) Weight. The weight of A is $\sum_{x \in A} F(x)$, the sum of the gray levels of the hexagons which it comprises. The weight is a measure of the intensity of the image throughout the area of the aggregate.
- 2) Gradient. The gradient of A is the GBT sum $\sum_{x \in A} F(x) \cdot \vec{x}$ where \vec{x} is the GBT offset vector for x relative to the center of A. The gradient shows the average direction and intensity of change across the aggregate.
- 3) Orientation. The orientation of A is a vector parallel to the gradient and scaled in length so that, when added to the address of the aggregate center, it gives the address of the adjacent aggregate in the direction of the gradient.
- 4) Centroid. The centroid or center of mass of A is its gradient divided by its weight.
- 5) Pattern type. The seven subordinates of the aggregate A can be labeled A_0, A_1, \dots, A_6 where A_0 is the aggregate one level down in the center or 0 position in A, A_1 is in the 1's position, etc. If T is a binary function on the set of all hexagon and aggregate GBT addresses, define the pattern type of A (with respect to T) to be $\sum_{i=0}^6 T(A_i) \cdot 2^i$, a number in the interval from 0 to 127. This defines pattern type in general as a function of the mapping T. Various T functions were used in the software developed during this research. Each involved thresholding the weight of an aggregate, and classifying it as "set" or "unset" for the purpose of pattern type. Table 1 shows all 128 pattern types.

We can illustrate the classification of an aggregate by pattern type using the level three aggregate A outlined in Figure 1. Here the picture function F is binary. A dot indicates a pixel which is set. Define the thresholding function T so that $T(B)=1$ if and only if more than half the hexagons in B are set. For this function T, the 0, 1, 3, 4, and 6 subordinates are unset while 2 and 5 are set. This corresponds to pattern type 0100100, number 36 in Table 1.

Once the aggregates have been assigned a pattern type, the pattern types are categorized. In general these categories help the algorithm decide whether the pixels in a given aggregate can be converted to a single point in the vector form. For aggregate A in Figure 1 such a decision would be bad since the aggregate intersects two lines which should not be merged. On the other hand, the pixels in aggregate 277 could be represented by a single point. Aggregate A is an example of a pattern type (#36) classified as "pathological" for the line-following algorithm. The pathologicals are those types which seem to contain parts of more than one line or structure in the picture. Other pathological types are 6, 18, 20, 22, 24, 26, 28, 30 and 38 in Table 1. The categories for line-following, besides pathological are: full, empty, edge, and transit. Pattern types 126 and 127 are classified as full while numbers 0 and 1 are empty. Edge types are those that appear to contain one edge of a line, for example types 2, 3, 14, and 15.

A transit aggregate seems to have a line running through it. Pattern types 21, 25, and 27 are examples of transit cells.

These categories have symmetry with respect to complementation. Each full type is the complement of an empty type, each pathological is the complement of a transit, and the complement of an edge type is again an edge type.

For the edge-following algorithm, pattern types are classified into four groups: full, empty, edge, and pathological. The first three are identical to those categories for line-following. For edge-following, however, transit cells are grouped with the pathologicals.

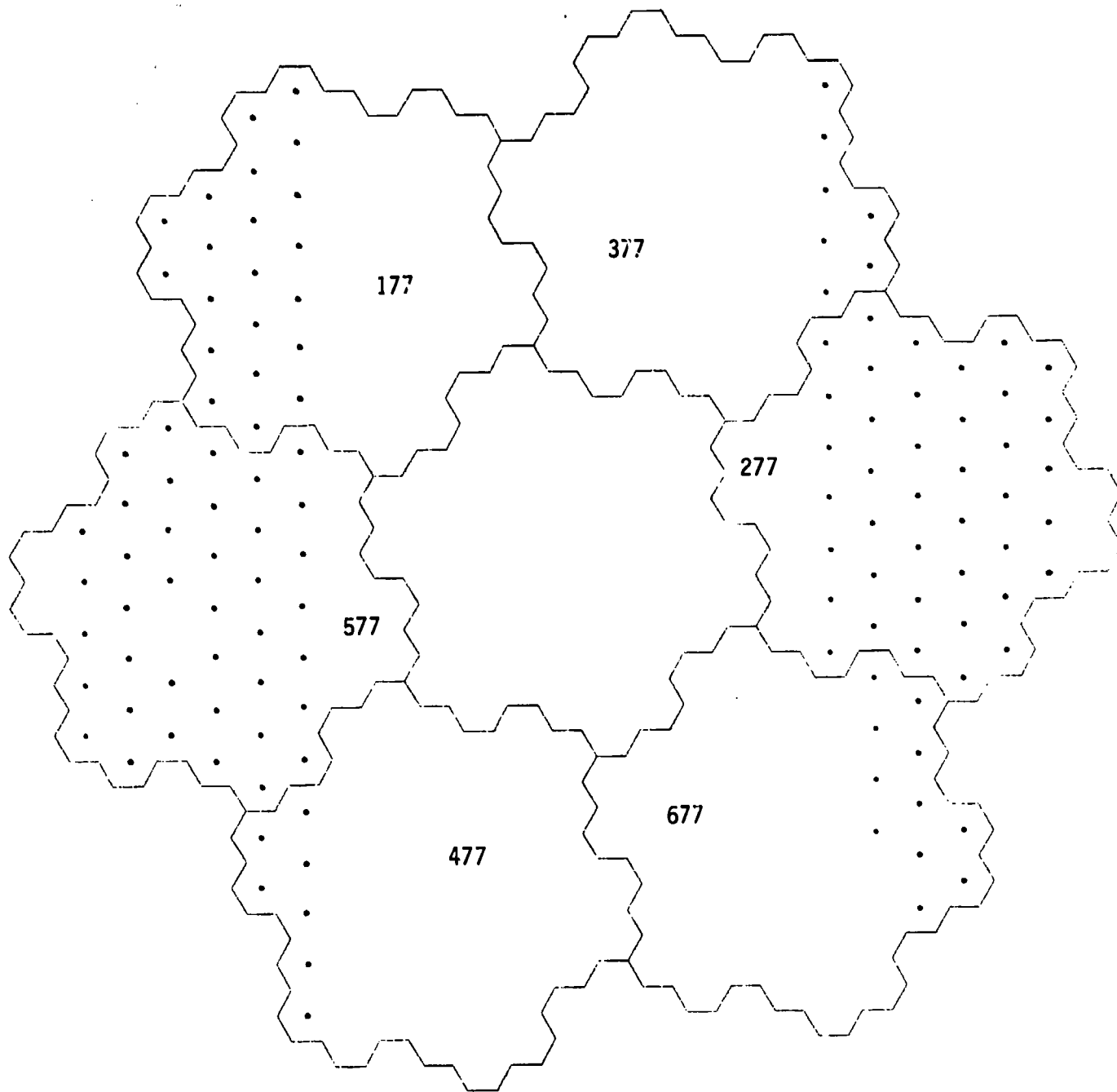


FIGURE 1: The Level Three Aggregate A has a Pathological Pattern Type.

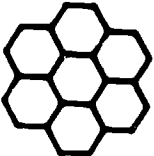
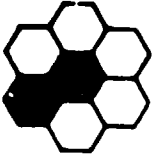
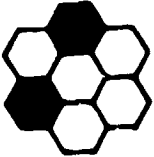
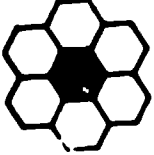
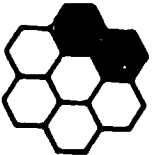
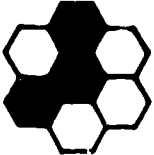
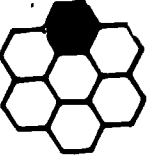


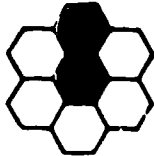
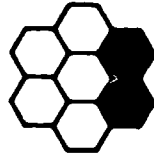
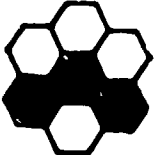
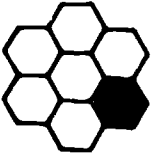
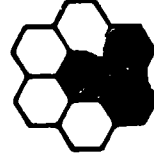

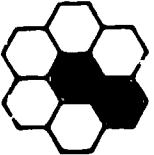
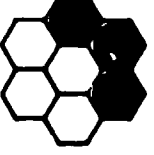
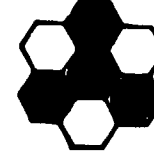
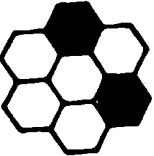
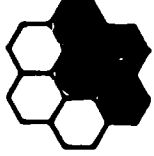

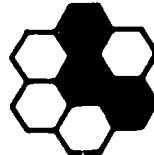
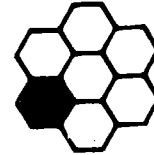


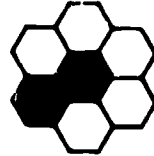

0000000 0		0001001 9		0010010 18	
0000001 1		0001010 10		0010011 19	
0000010 2		0001011 11		0010100 20	
0000011 3		0001100 12		0010101 21	
0000100 4		0001101 13		0010110 22	
0000101 5		0001110 14		0010111 23	
0000110 6		0001111 15		0011000 24	
0000111 7		0010000 16		0011001 25	
0001000 8		0010001 17		0011010 26	

TABLE 1: 128 Pattern types

0011011 27		0100100 36		0101101 45	
0011100 28		0011100 37		0100101 46	
0011101 29		0100110 38		0101111 47	
0011110 30		0100111 39		0110001 48	
0011111 31		0101000 40		0110001 49	
0100000 32		0101001 41		0110010 50	
0100001 33		0101010 42		0110011 51	
0100010 34		0101011 43		0110100 52	
0100011 35		0101100 44		0110101 53	

TABLE 1: 128 Pattern Types

0110110 54		0111111 63		1001000 72	
0110111 55		1000000 64		1001001 73	
0111000 56		1000001 65		1001010 74	
0111001 57		1000010 66		1001011 75	
0111010 58		1000011 67		1001100 76	
0111011 59		1000100 68		1001101 77	
0111100 60		1000101 69		1001110 78	
0111101 61		1000110 70		1001111 79	
0111110 62		1000111 71		1010000 80	

TABLE 1: 128 Pattern Types



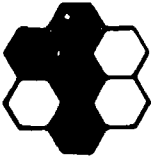
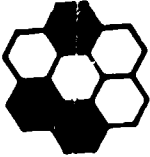
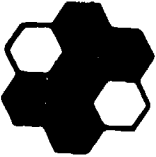

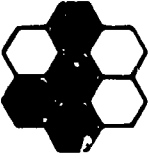

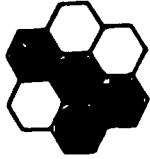


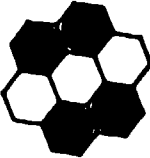


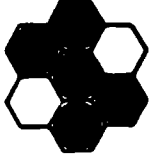


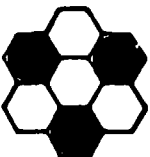

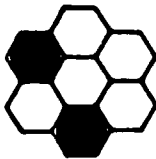
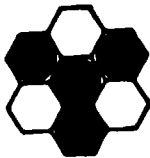
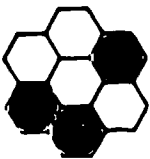
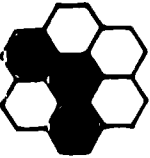

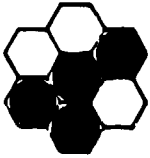


1010001 81		1011010 90		1100011 99	
1010010 82		1011011 91		1100100 100	
1010011 83		1011100 92		1100101 101	
1010100 84		1011101 93		1100110 102	
1010101 85		1011110 94		1100111 103	
1010110 86		1011111 95		1101000 104	
1010111 87		1100000 96		1101001 105	
1011000 88		1100001 97		1101010 106	
1011001 89		1100010 98		1101011 107	

TABLE 1: 128 Pattern Types

1101100 108		1110011 115		1111010 122	
1101101 109		1110100 116		1111011 123	
1101110 110		1110101 117		1111100 124	
1101111 111		1110110 118		1111101 125	
1110000 112		1110111 119		1111110 126	
1110001 113		1111000 120		1111111 127	
1110010 114		1111001 121			

TABLE 1: 128 Pattern Types

If it is assumed, as it was in Figure 1, that the picture function is binary and a dot indicates a pixel which is set, Figures 2 through 9 illustrate the aggregate descriptors. Figure 2 represents the raster form. Figures 3 and 4 show weighted centroids for level two and three aggregates, respectively. By weighted centroid we mean that the center of the hexagon is the aggregate centroid as defined in (4) above and that the size of the hexagon is proportional to the aggregate weight. Figure 5 shows level two gradients while Figures 6 and 7 illustrate level two and level three orientations. Figures 8 and 9 show pattern types (for line-following) at levels two and three, respectively. The line segment indicates an edge type, a triangle is a transit, and a diamond is a pathological.

4.1.2 The Line-Following Algorithm

In line-following or centerlining, we are looking for pairs of edge aggregates that contain opposite edges of the line or for transit aggregates which appear to have the line passing through them. The algorithm tries to work with aggregates at a fixed level. It seeks the level at which edge and transit types predominate. If the level is too low, full cells will be numerous. If it is too high, there will be many pathologies.

Line following begins by choosing an edge type aggregate. This cell's orientation is added to its address to obtain the address of the cell which most likely contains the opposite edge of the line. If this second cell is empty, the algorithm concludes that the line center corresponds to the centroid of the first cell; if the second cell is of the edge type and its orientation points back to the first cell, the algorithm concludes that the weighted average of the centroids of these two cells is the line center (See Figure 10). If the second cell does not point back but does not point directly away from the first cell then a chain of aggregates exists, having the property that the first points to the second, the second to a third, and so on. These chains are allowed to grow to some length, n , and then the weighted average of all the centroids is taken. See Figure 11 for an example of such a chain.

Transit cells are handled in a different way. Since a transit seems to contain the whole line, its centroid is assumed to be the line center. Figure 12 shows a transit cell and its centroid.

After a line center point (centroid or average of centroids) is computed, one of two mechanisms is used to find the next line center point. For an edge cell, its orientation is rotated 60 degrees by (GBT) multiplying the orientation vector by 3 or 5. The rotated vector is added to the cell address to yield the address of an aggregate which is a good candidate to contain the next piece of the line. For transit cells the orientation vector can be misleading. In this case pattern type is used. The pattern type of a transit aggregate tells which way the line passes through it. For example, for pattern type 100011, number 67, the line moves from the 1 direction to the 6 direction. For number 97, 1100001, the line moves from the 5 direction through the center to the 6 direction. Thus for an aggregate with pattern type 97, if the previous line center point is out in the 5 direction, the next line center point will most likely be out in the 6 direction.

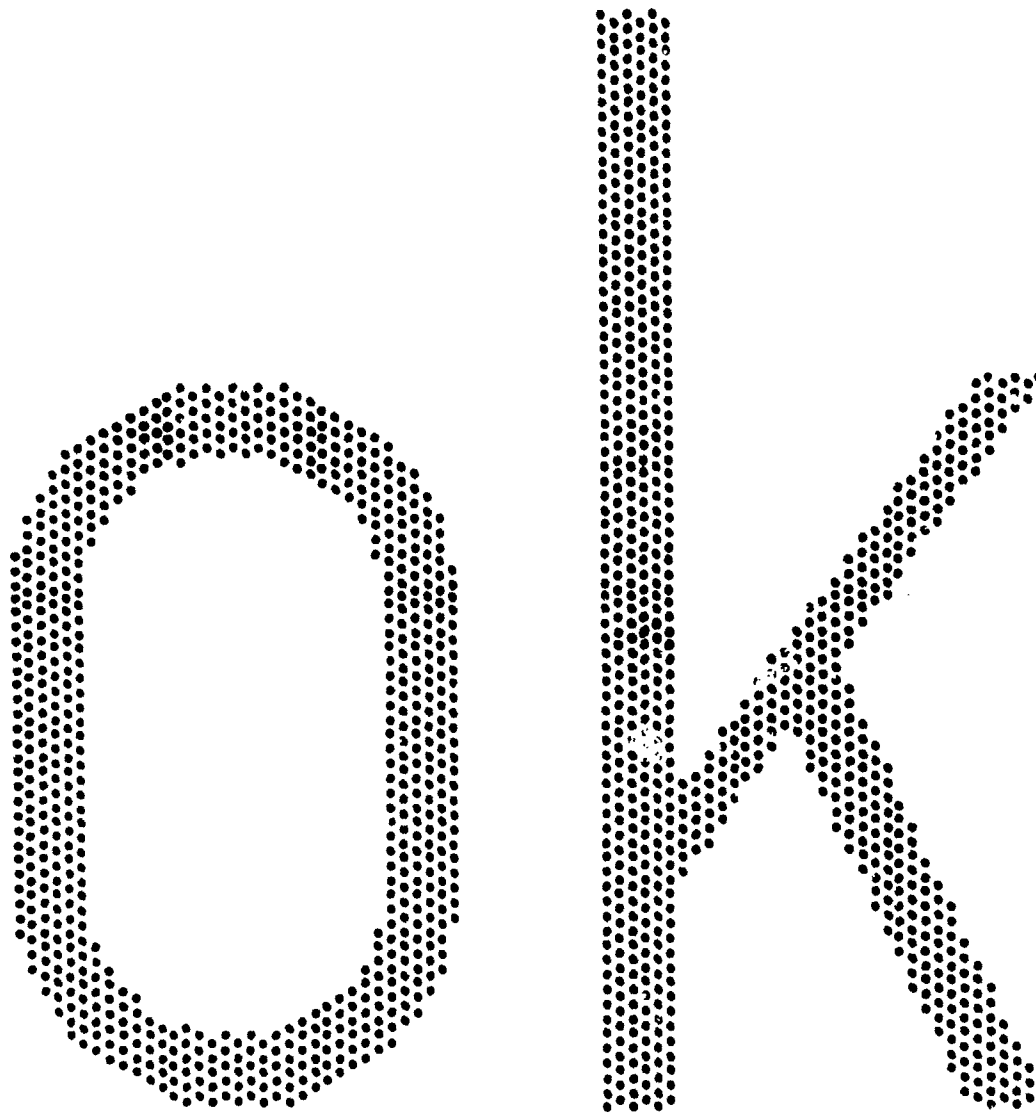


FIGURE 2: A Raster Scan of the Letters "OK"

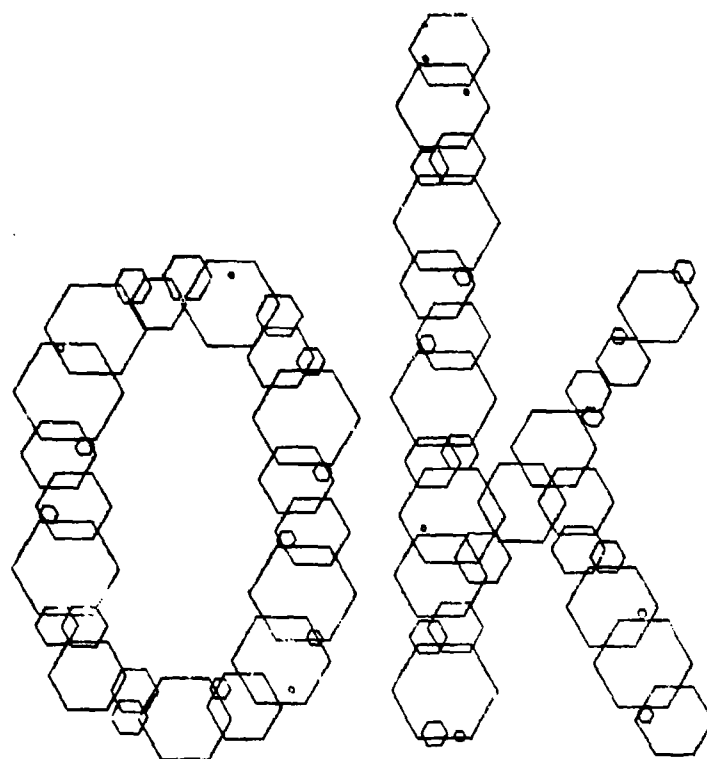


FIGURE 3: Level Two Weighted Centroids for "OK"

The aggregate centroid is at the center of each hexagon. The weight of the aggregate is reflected by the size of the hexagon.

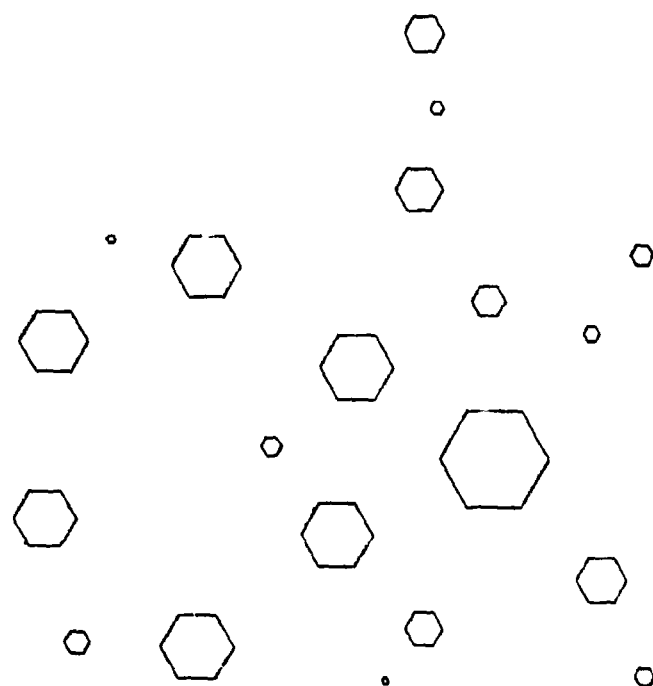


FIGURE 4: Level Three Weighted Centroids for "OK"

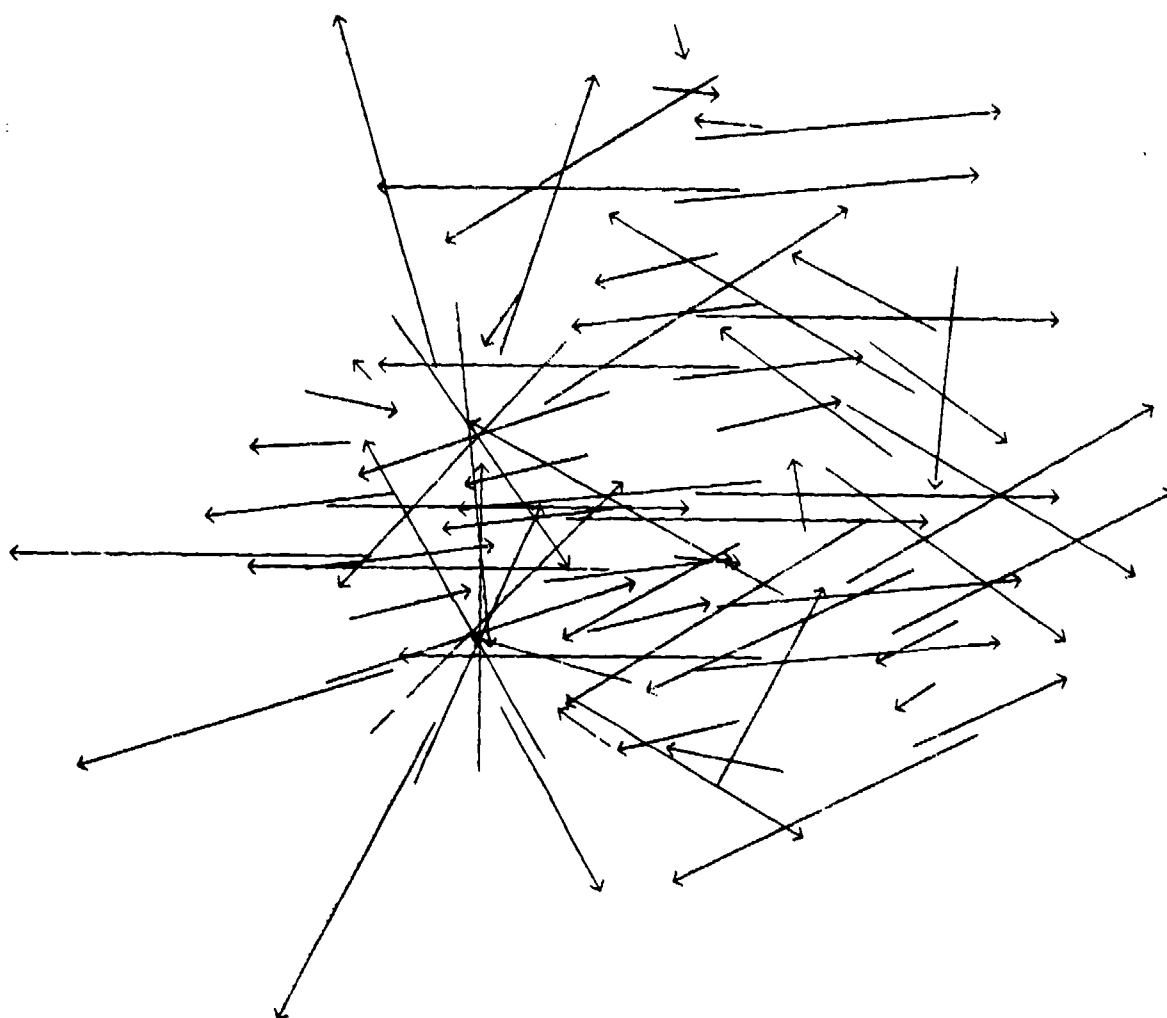


FIGURE 5: Level Two Gradients for "OK"

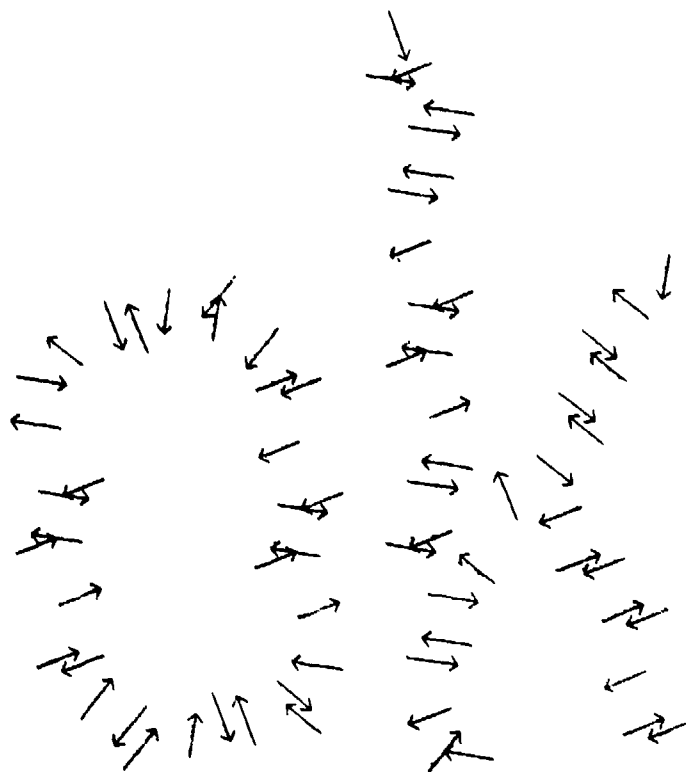


FIGURE 6: Level Two Orientations for "OK"

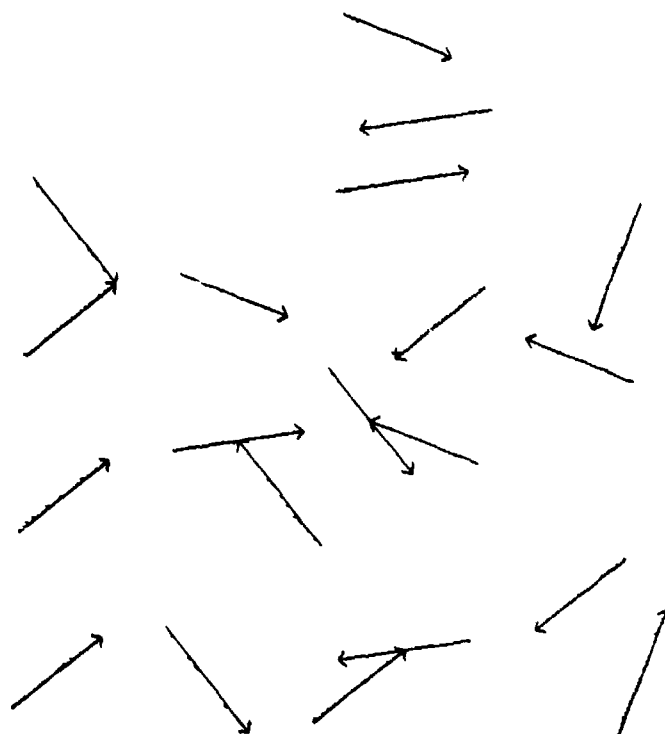


FIGURE 7: Level Three Orientations for "OK"

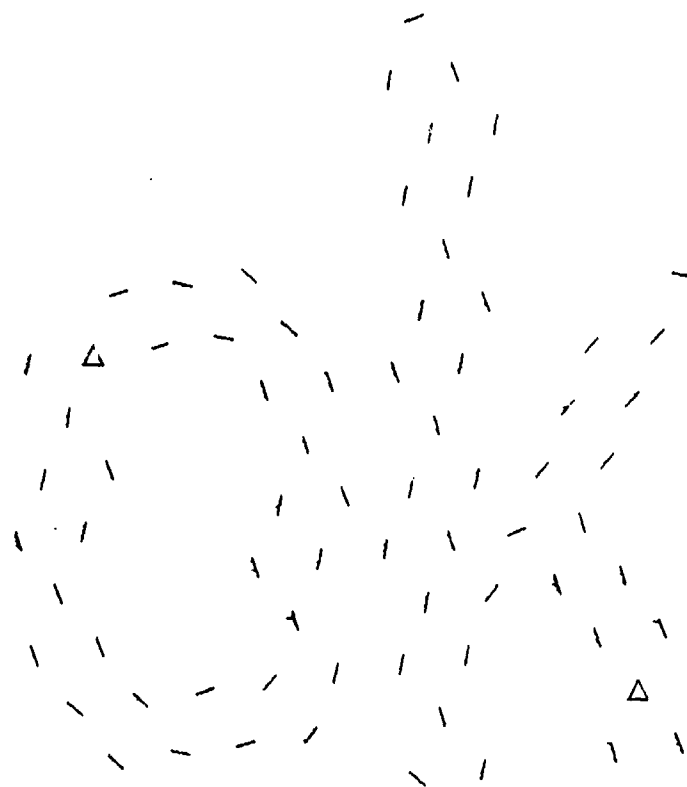


FIGURE 8: Line-Following Pattern Types for Level Two Aggregates

A line segment indicates an edge cell, a triangle is a transit.

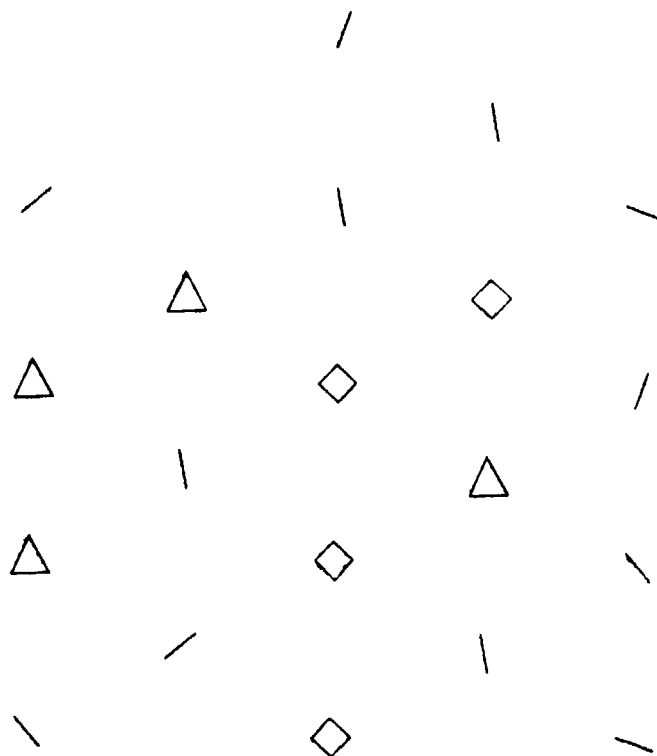


FIGURE 9: Line-Following Pattern Types for Level Three Aggregates
A diamond represents a pathological cell.

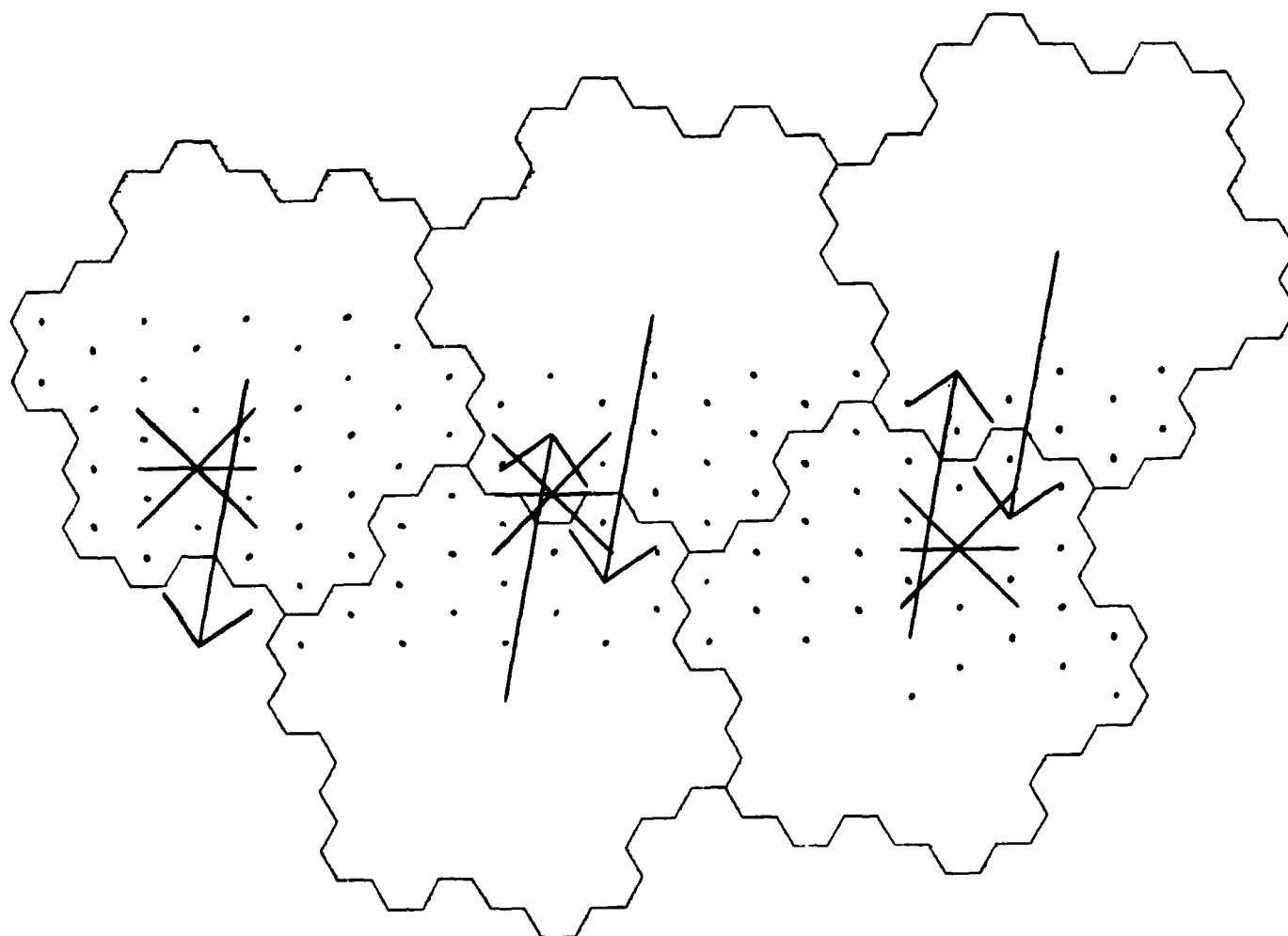


FIGURE 10: Edge cells are Paired by Their Orientations.

The large asterisks indicate the combined centroids.

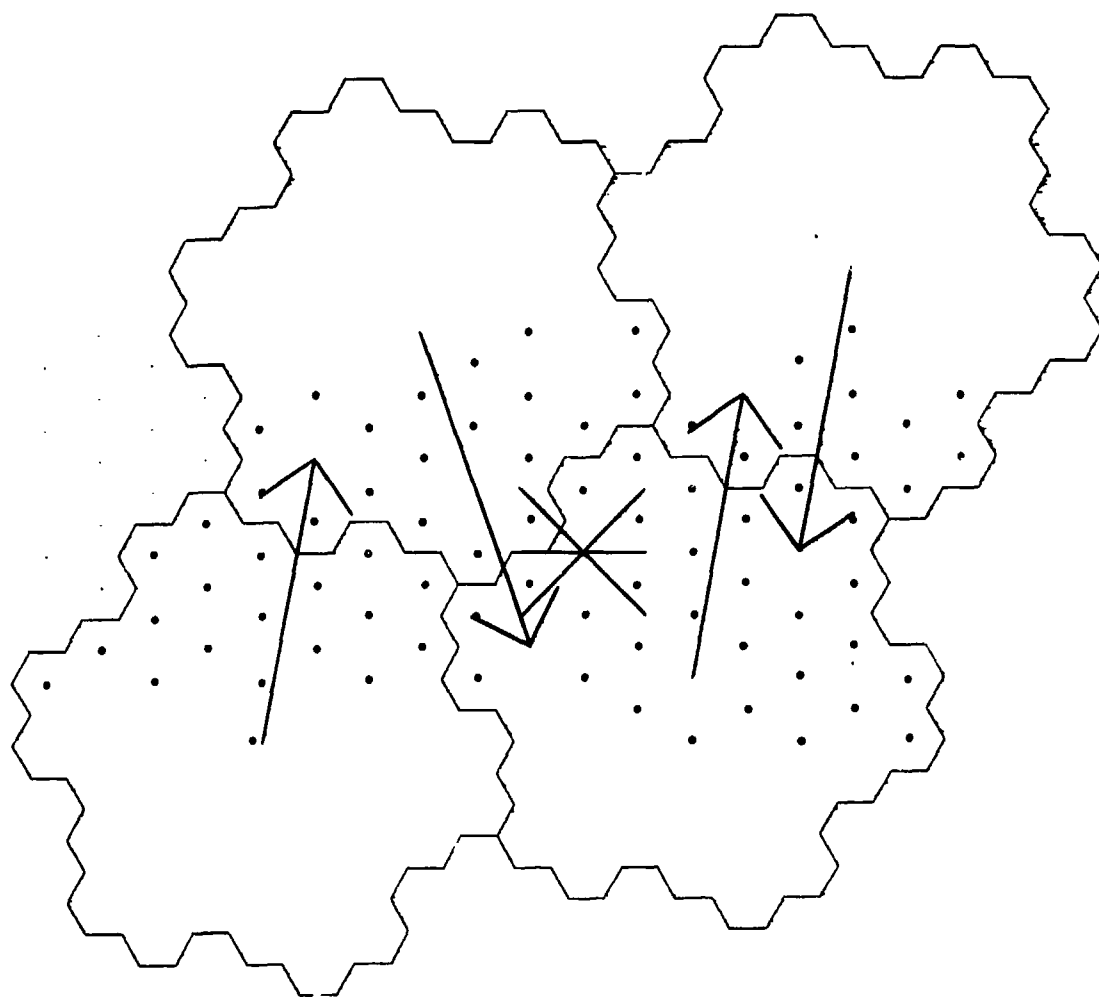


FIGURE 11: For this Chain of Four Cells a Weighted Centroid is Computed.

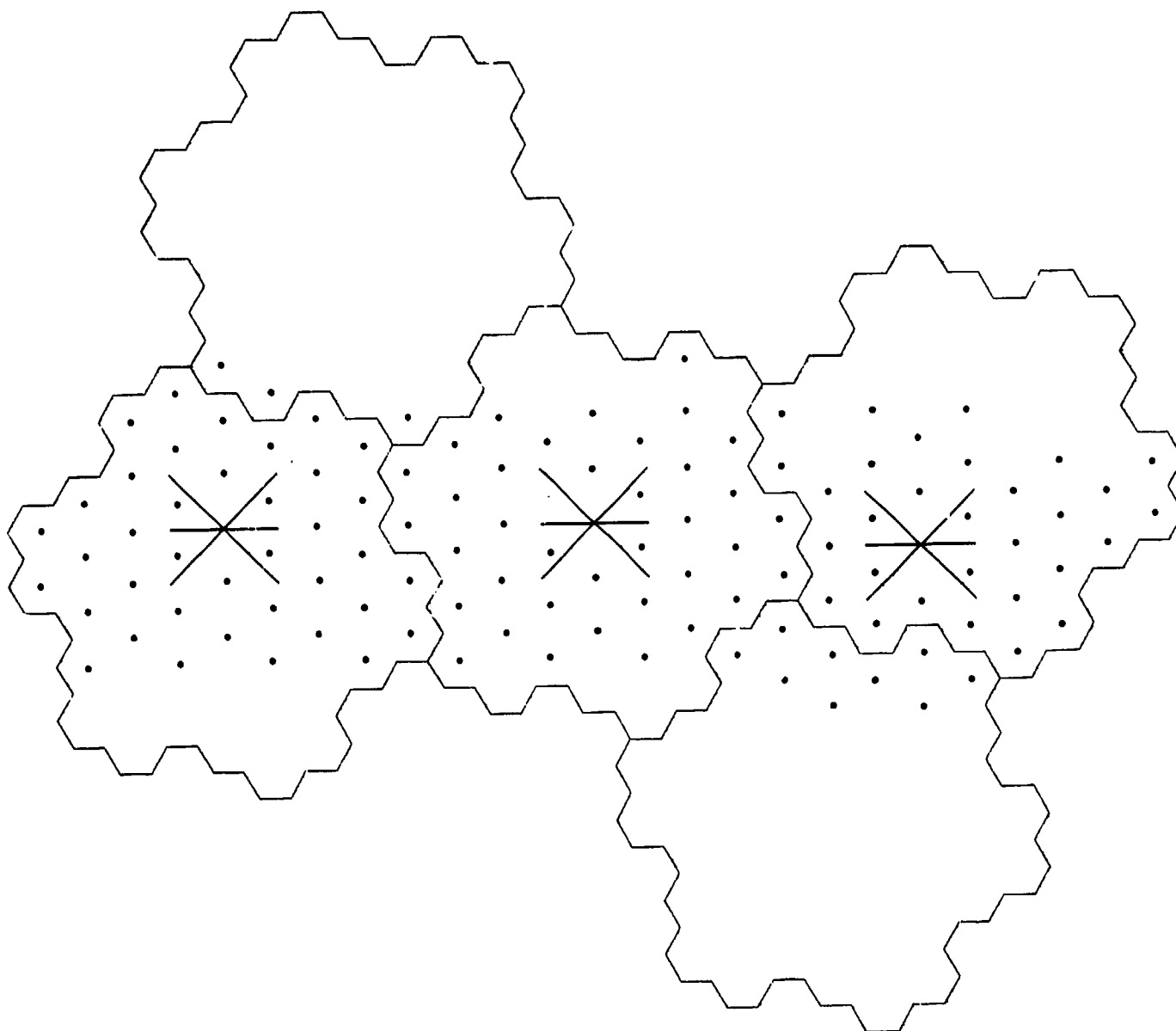


FIGURE 12: A Transit Cell (center) and its Centroid.

Pathological aggregates provide the mechanism for changing levels.

Since a pathological cell seems to contain parts of different lines or structures, the bits within such a cell should not be averaged into one centroid. When the algorithm reaches a pathological aggregate, it moves down one level in the hierarchy and considers it only in terms of its seven constituent cells. For example, in Figure 13, aggregate B points to the pathological aggregate A. Within A, the cell labeled 377 points back to B. The centroid for the combination of 377 and B is computed. By examining the pattern type of A, the algorithm determines the next aggregate to be processed.

4.1.3 The Boundary-Following Algorithm

The boundary-following algorithm is simpler than the one for center-lining. There are fewer categories of pattern types and so, fewer cases to be considered. A different technique is used to average the pixels within an aggregate. Instead of the centroid, a boundary centroid is computed in this way. Assume a binary picture function F and recall that the centroid of a (non-empty) aggregate A is its gradient, (GBT) $\sum_{x \in A} F(x) \cdot \vec{x}$ (where \vec{x} is the GBT offset vector for x relative to the center of A), divided by its weight, $\sum_{x \in A} F(x)$. Define the anti-centroid for a non-full aggregate as (GBT) $\sum_{x \in A} (1-F(x)) \cdot \vec{x}$ divided by $\sum_{x \in A} (1-F(x))$. Then the boundary centroid is the GBT sum of the centroid and the anti-centroid. Figure 14 shows the centroid, the anti-centroid, and the boundary centroid. The centroid is the center of mass of the set hexagons while the anti-centroid is the center of mass of those that are unset. The boundary centroid is a point which is approximately on the boundary.

The boundary-following algorithm moves clockwise beginning with an edge aggregate. The boundary centroid of the aggregate is computed. The pattern type of the aggregate is used to find the aggregate which should contain the next piece of boundary. The algorithm looks for edge aggregates. If a pathological cell is reached, it is handled much as in line-following. It is considered only in terms of its seven constituent subaggregates. The algorithm continues until it has processed all edge cells.

4.1.4 The Smoothing Algorithm

Both line and boundary-following produce a large number of vectors. In order to reduce this vector volume the smoothing algorithm attempts to remove extraneous or redundant vectors. A second effect is to smooth out small zig-zags along lines.

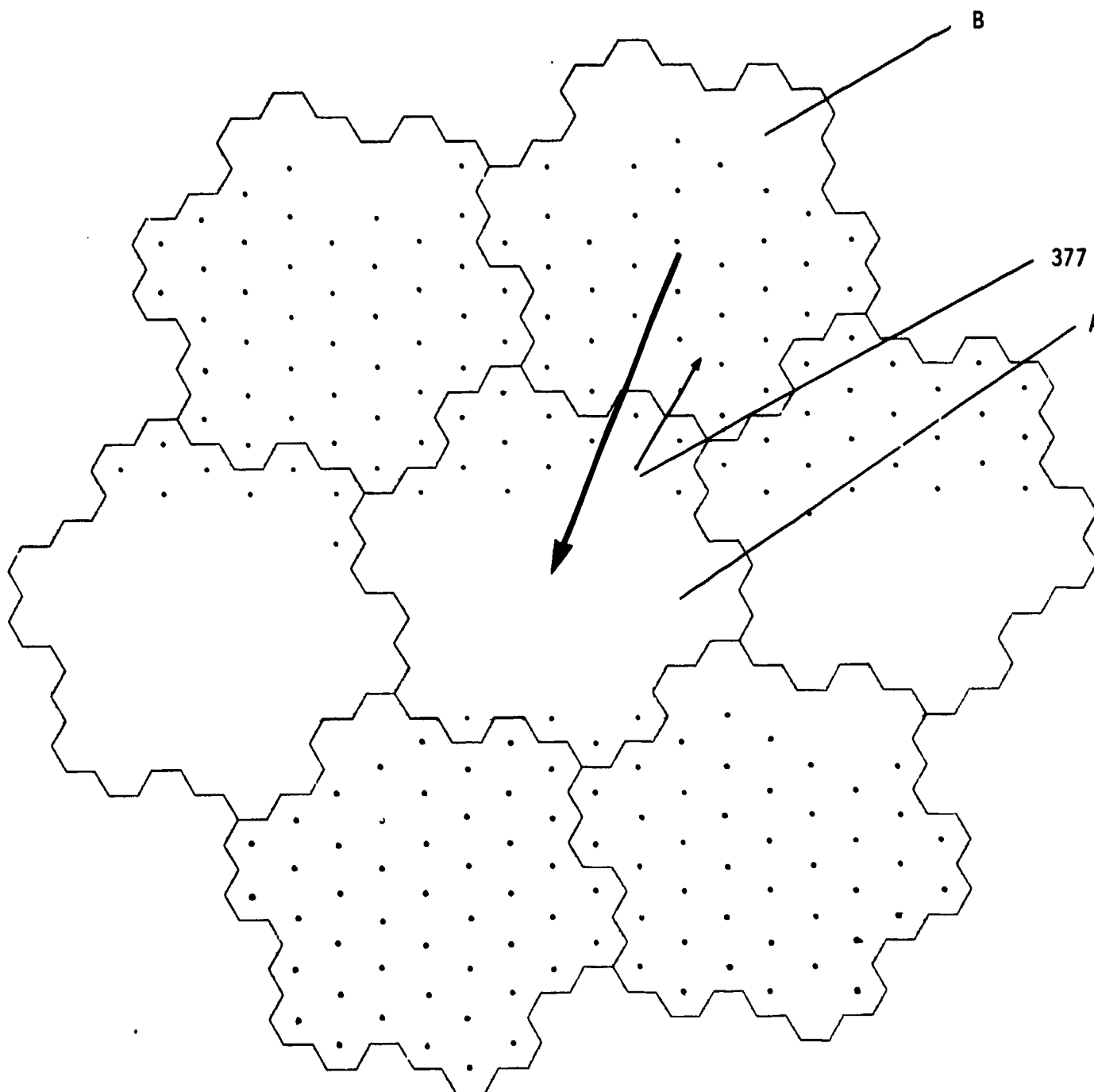


FIGURE 13: A is a Pathological Aggregate.

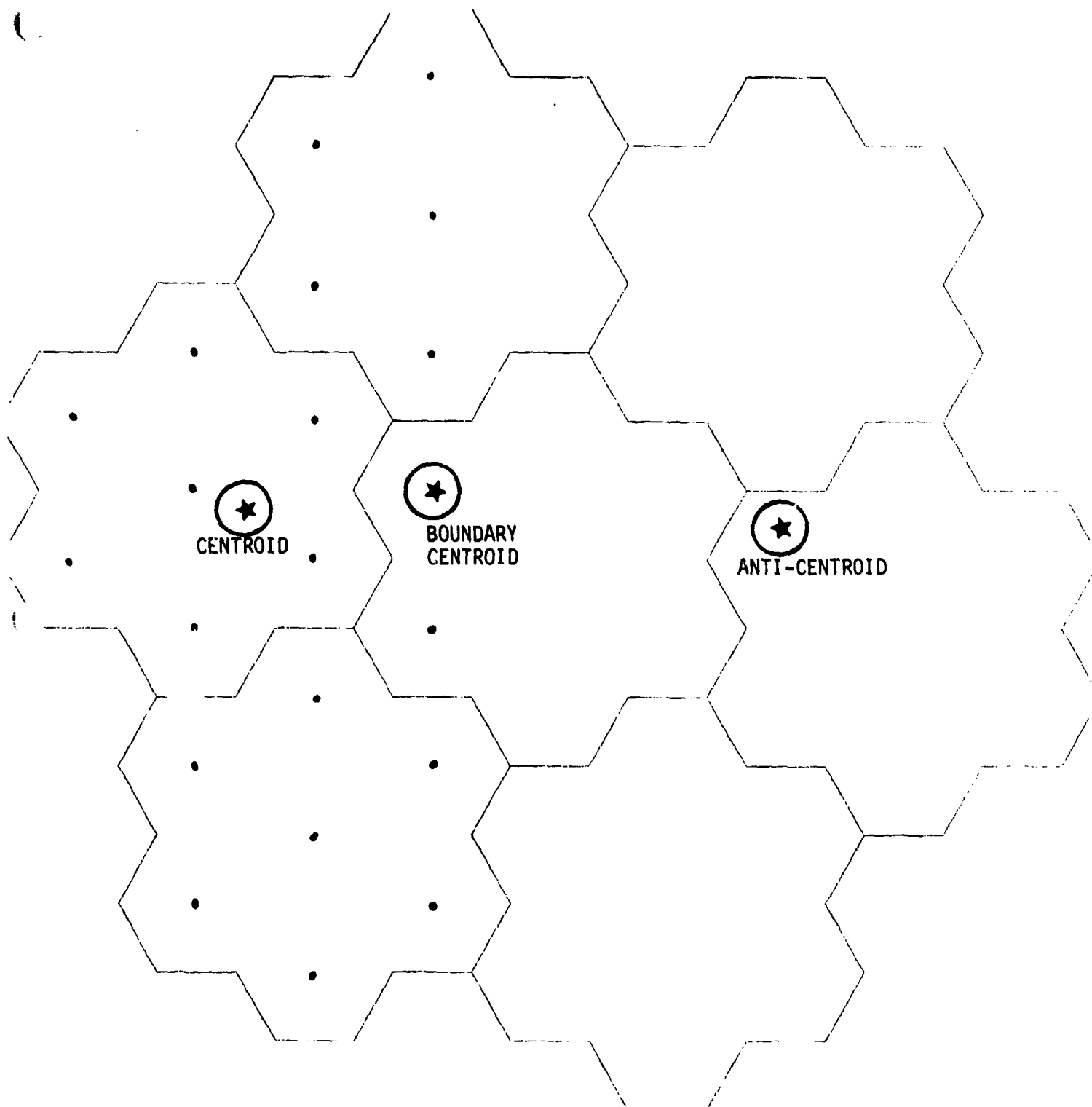


FIGURE 14: The Centroid, Anticentroid, and Boundary Centroid of an Aggregate.

In order to develop efficient means of smoothing, a function called the angle difference measure (ADM) is defined. The ADM is a GBT function which does not use any of the trigonometric functions. It assigns a number in the interval from -1 to 1 to the angle between two GBT vectors. For the purpose of this algorithm, $ADM(v_1, v_2)$ where v_1 and v_2 are GBT addresses, indicates whether or not the vectors from 0 to v_1 and from 0 to v_2 point in approximately the same direction.

Another concept important in smoothing is that of the degree of a point. A point in the vectorized form has degree n , where n is a non-negative integer, whenever p is connected to n other points. Thus line ends have degree 1, the points along a single line have degree 2, and a "T" intersection point has degree 3.

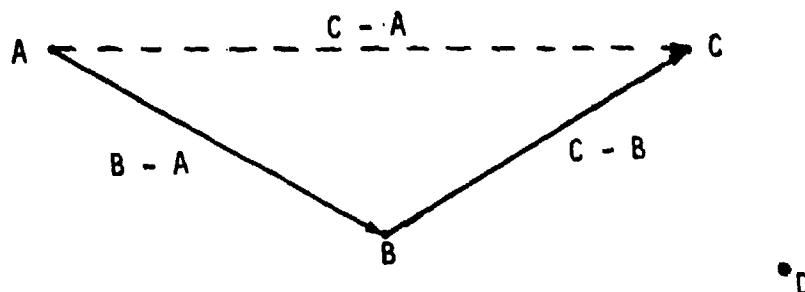
The smoothing algorithm processes only points of degree 1 and 2. It begins with a point of degree 1, if there are any. If there are none, it starts with a degree 2 point. Consider a start point A (see Figure 15). A is connected by a vector (B-A) to the point B. If B has degree 2, then B is connected to a third point, C, by a vector (C-B). The algorithm decides if the vectors B-A and C-B can be replaced by a vector C-A and the point B removed. Three parameters are used by the algorithm. The first, δ , is a notion of closeness of angles. The algorithm checks whether B-A and C-B point in approximately the same direction by comparing $ADM(B-A, C-B)$ and δ . If B-A and C-B pass this test, then the parameter max vector length is checked. Is the new vector C-A shorter than max vector length or is it too long? The third parameter is called cumulative error, CE. The formula for CE is: $CE = CE + ADM(B-A, C-A)$, where CE is initialized to zero. If all three conditions are satisfied then B is deleted. The next step is to decide whether or not C can be deleted. If C has degree 2 then there is a point D not equal to B connected to C. The algorithm wants to replace C-A and D-C with D-A. It compares $ADM(C-A, D-C)$ and δ and checks that D-A is not too long. It adds $ADM(C-A, D-A)$ to CE and checks that this error is within the given bound.

The algorithm continues until all points of degree 1 or 2 are processed.

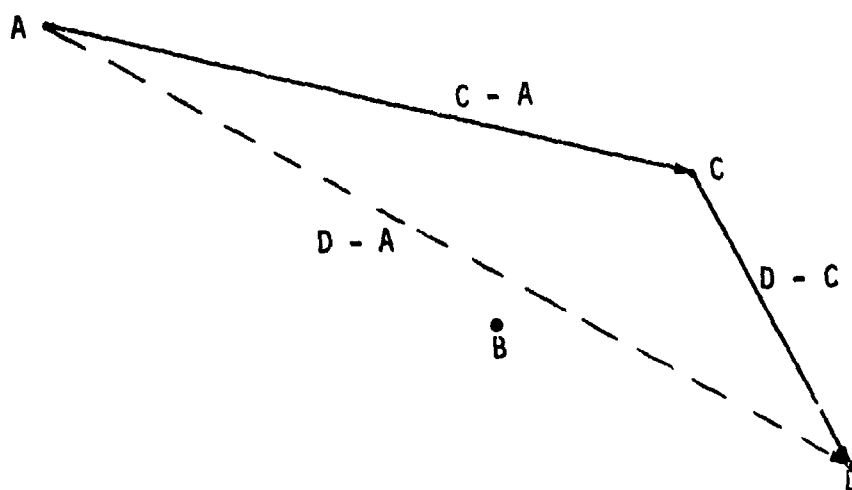
4.2 Generalized Balanced Ternary

4.2.1 The Structure and Addresses

The GBT structure is one of a hierarchy of cells. At each level, the cells are constructed of cells from the previous level according to a rule of aggregation. The basic cells of this structure are hexagons. Figure 16 shows the hexagonal covering of the plane. This covering has the uniform adjacency property, that is, each element of the covering is adjacent to exactly six other elements and shares with each exactly one-sixth of its boundary. In contrast, a covering of the plane with squares does not have uniform adjacencies. Some squares are adjacent at a point while others share a side.



Step 1. Replace vectors $B-A$ and $C-B$ with $C-A$



Step 2. Replace vectors $C-A$ and $D-C$ with $D-A$

FIGURE 15: Smoothing the line ABCD

A first level aggregate is formed by taking a single hexagon and its six neighbors (see Figure 17A). The first level aggregates also cover the plane and have the uniform adjacency property. In general, an aggregate at level n is formed by taking a level $n-1$ aggregate and its six neighbors. It can be shown that the planar covering and uniform adjacency properties hold at each level. Figures 17B and 17C show second and third level aggregates.

The GBT addressing system is based on the following scheme: In an aggregate, the center cell is labeled 0 and the outer six cells are labeled, in clockwise order, 1, 3, 2, 6, 4, 5. (See Figure 17). Each hexagon in the plane has a unique GBT address, a sequence of digits corresponding to the labels of the cells above that hexagon.

Each digit of the address corresponds to an aggregate level. For example, the address 536 labels the hexagon in the 6 position of the first level aggregate, which is in the 3 position of the second level aggregate, which is in the 5 position of the third level aggregate, which is at the 0 or center position at all higher levels. The hexagon 536 is shaded in Figure 18.

The digit 7 is used to address entire aggregates rather than hexagons. Thus, the first level aggregate shaded in Figure 19 has address 117. The second level aggregate outlined in the same figure has address 677.

The symbols 0, 1, 2, 3, 4, 5, 6, and 7, used as GBT digits, are also used in octal and decimal notation to express integers. Using them in this new context allows GBT addresses to be handled directly by computer hardware and software. Care must be taken, however, not to confuse the addresses with integers.

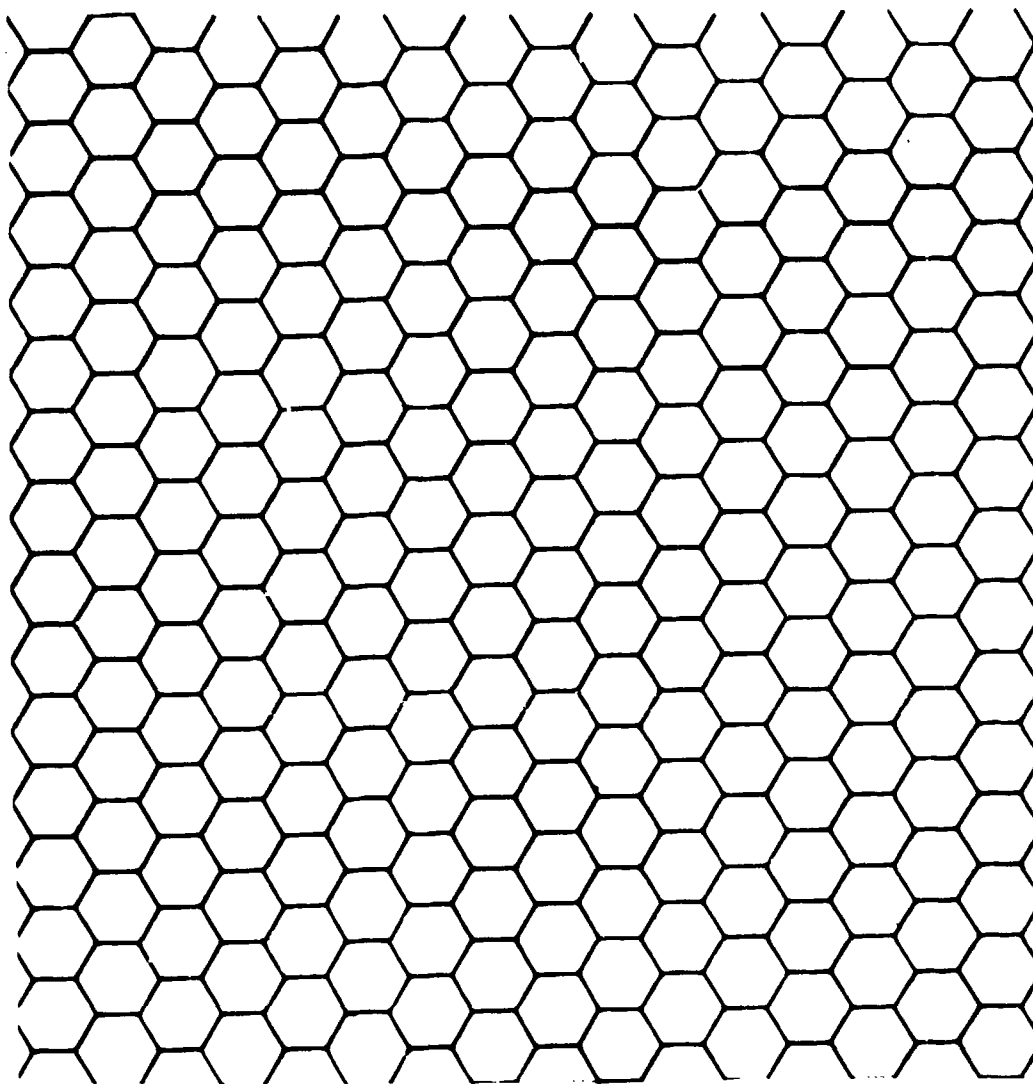


FIGURE 16: The Hexagonal Covering of the Plane

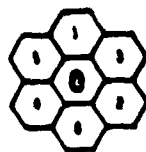


FIGURE 17A: First Level Aggregate

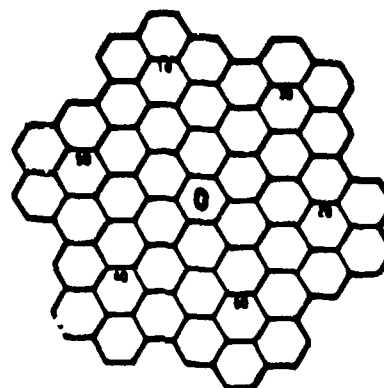


FIGURE 17B: Second Level Aggregate

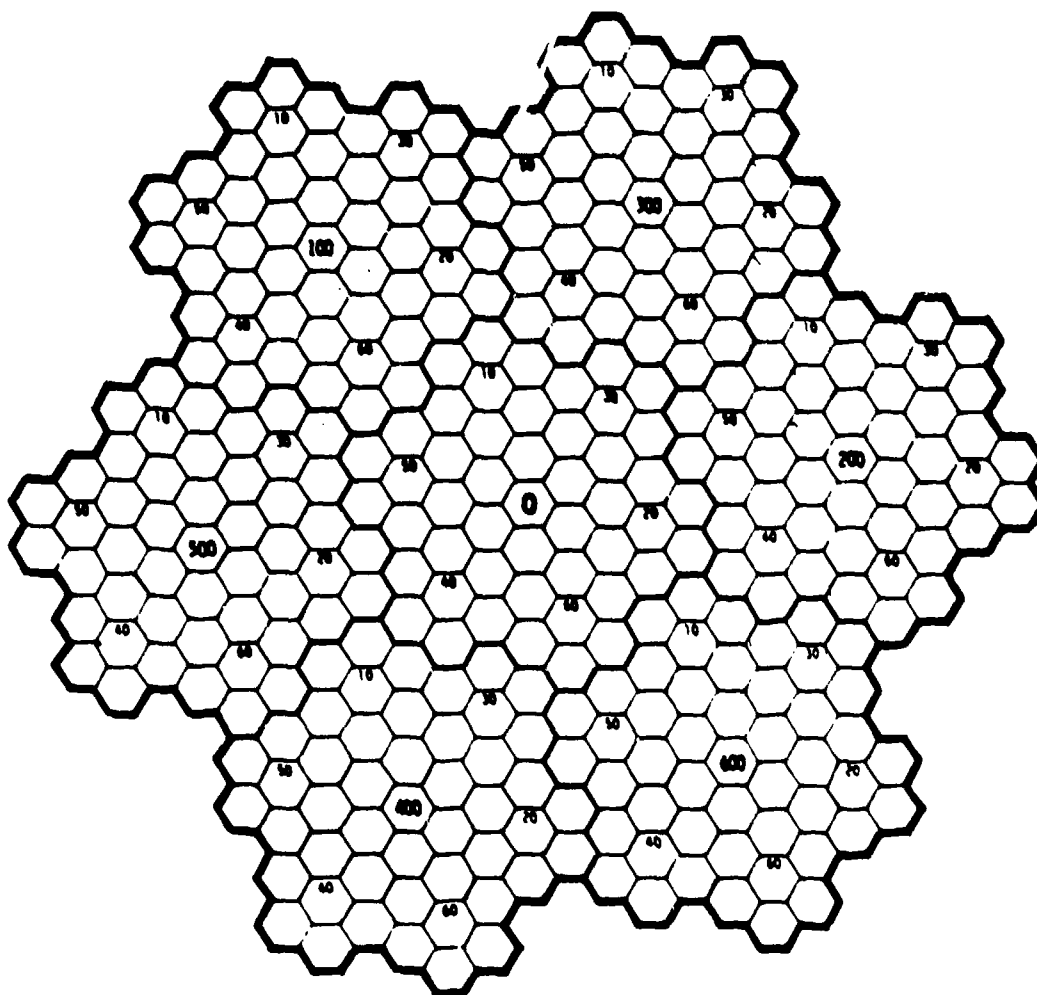


FIGURE 17C: Third Level Aggregate

FIGURE 17: The Aggregate Structure

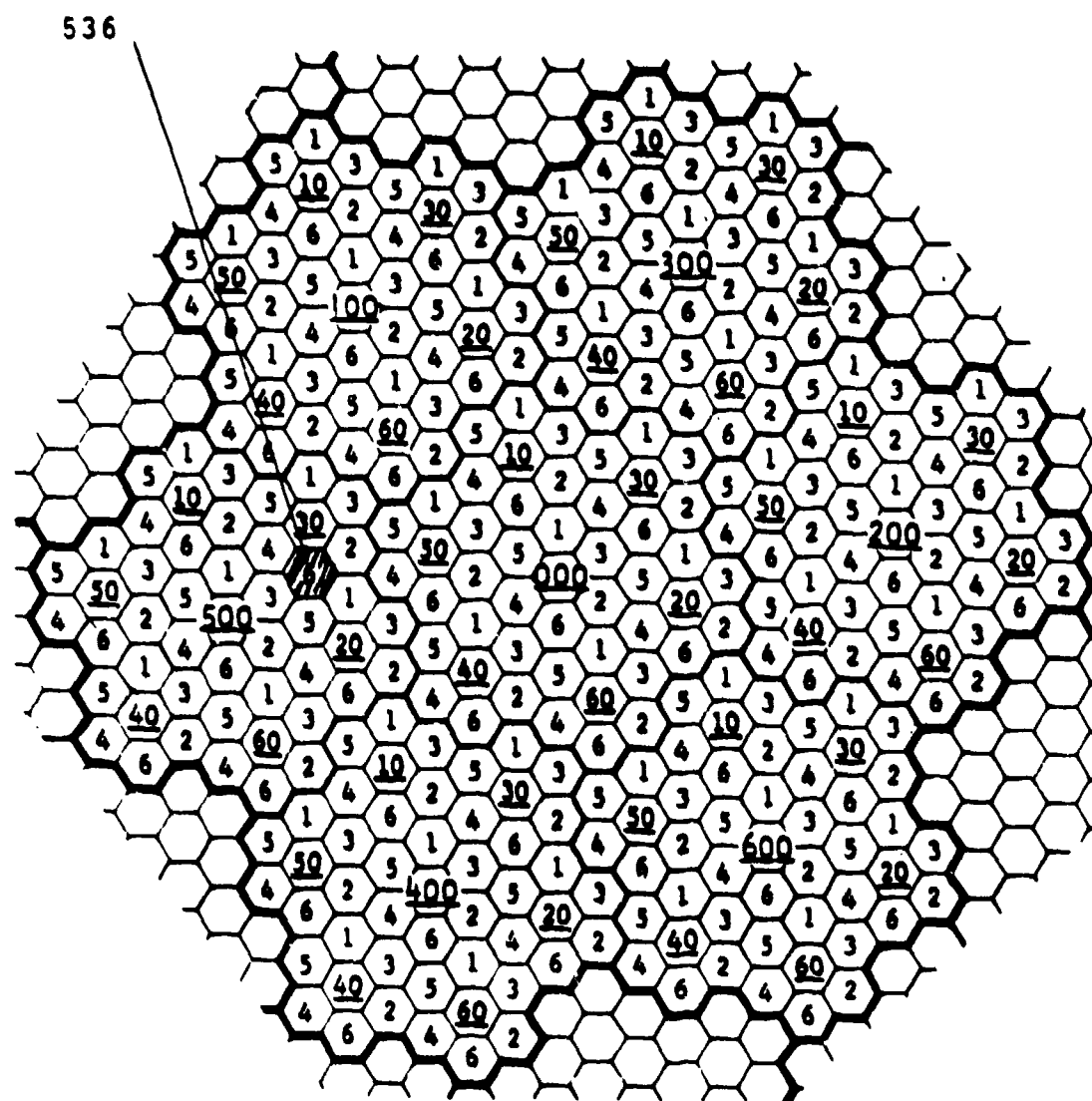


FIGURE 18: The Location of the Hexagon whose Address is 536

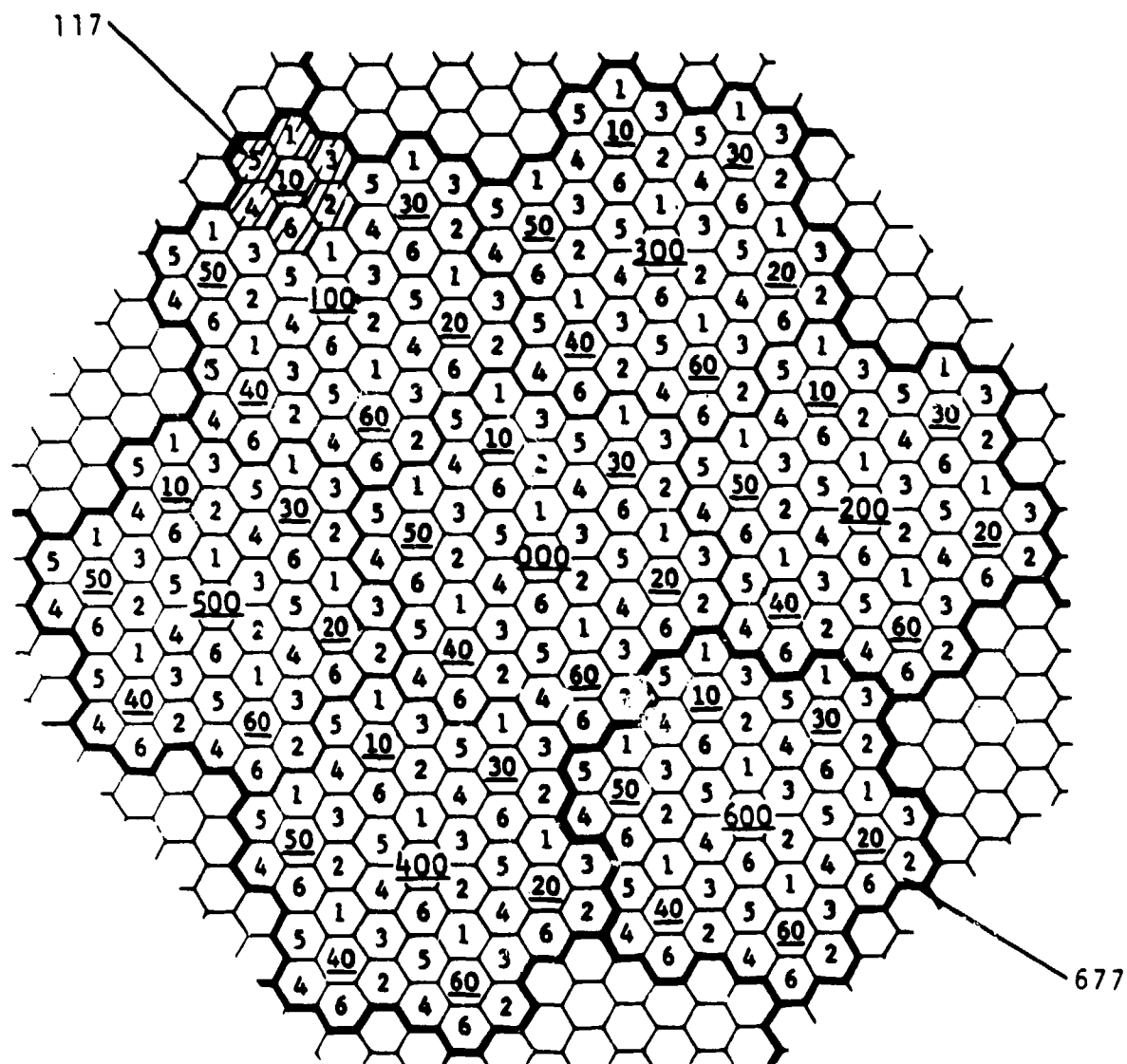


FIGURE 19: Two Examples of the Use of the Trailing Sevens Notation to Address Higher Aggregates

4.2.2 The Arithmetic

In order for the GBT addressing system to be useful, there must be efficient methods for doing planar addition, subtraction, and multiplication entirely in terms of GBT addresses. These methods are discussed below:

- 1) Addition: Addition of GBT addresses parallels integer addition in that if an addition table for the seven digits is given, any two addresses can be added. From Figure 20, we can derive the basic GBT addition table. Using standard planar parallelogram addition, we see, for example, that $1+2=3$; $3+6=2$, and $5+6=4$. If 3 and 2 are added, the sum is outside the central first level aggregate, $3+2=25$. Thus, Table 2 is obtained. Addition of multidigit addresses is very much like adding multidigit integers. For example (See Figure 21), to add 153 and 45, add $3+5=1$, then $5+4=52$, and carrying the 5 to the next column, $5+1=16$. Thus, $153+45=1621$. Figure 21B shows these vectors in the plane.
- 2) Subtraction: Subtraction is accomplished by the process of complementing and adding. The complement of a GBT address is its digitwise sevens complement. The complement of 61542 is 16235, for example. In Figure 20, we see that 3 and 4 are complements, as are 1 and 6, and 53 and 24.
- 3) Multiplication: GBT multiplication is similar to addition in that it is a digitwise operation. The multiplication table (Table 3) shows that the GBT product of two digits is just their integer product modulo 7. An example of multidigit multiplication is:

254 x 62:

$$\begin{array}{r} 254 \\ \times 62 \\ \hline 43T \\ 523 \\ \hline 526T \end{array} \quad \begin{array}{l} (= 2 \times 254) \\ (= 6 \times 254) \\ (= \text{the GBT sum}) \end{array}$$

Figure 22 illustrates this product geometrically. Notice that the angle in this figure is the sum of angles A and B. The length of the vector from 0 to 5261 is the product of the length of the vectors 254 and 62. Because GBT multiplication corresponds to complex multiplication, it can be used to rotate and stretch vectors. This property is used in the RVC line-following algorithm.

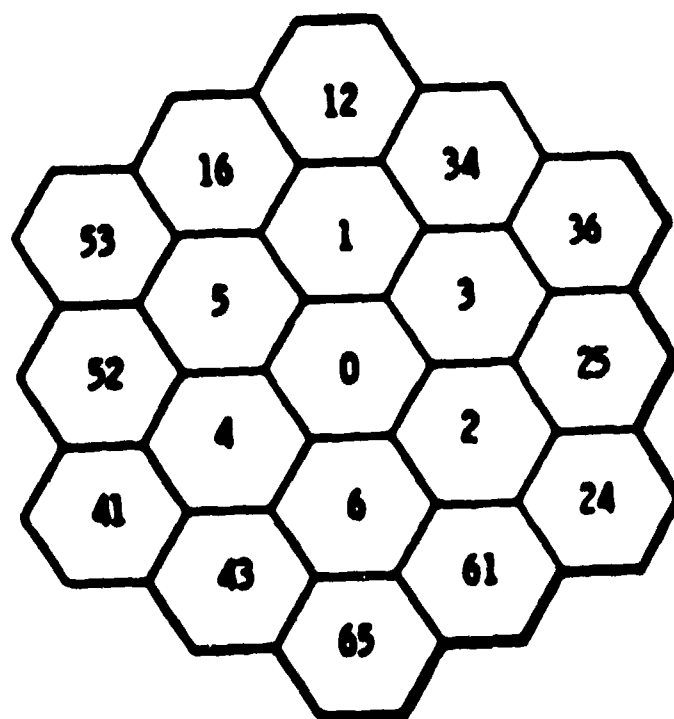


FIGURE 20: Key to Planar Addition Table

+	0	1	2	3	4	5	6
0	0	1	2	3	4	5	6
1	1	12	3	34	5	16	0
2	2	3	24	25	6	0	61
3	3	34	25	36	0	1	2
4	4	5	6	0	41	52	43
5	5	16	0	1	52	53	4
6	6	0	61	2	43	4	65

TABLE 2: GBT Addition

FIGURE 21A

Add the Addresses 153 and 45:

$$\begin{array}{r} (1)(5) \\ 153 \\ 45 \\ \hline 1621 \end{array} \quad () \text{ Denotes a Carry}$$

FIGURE 21B

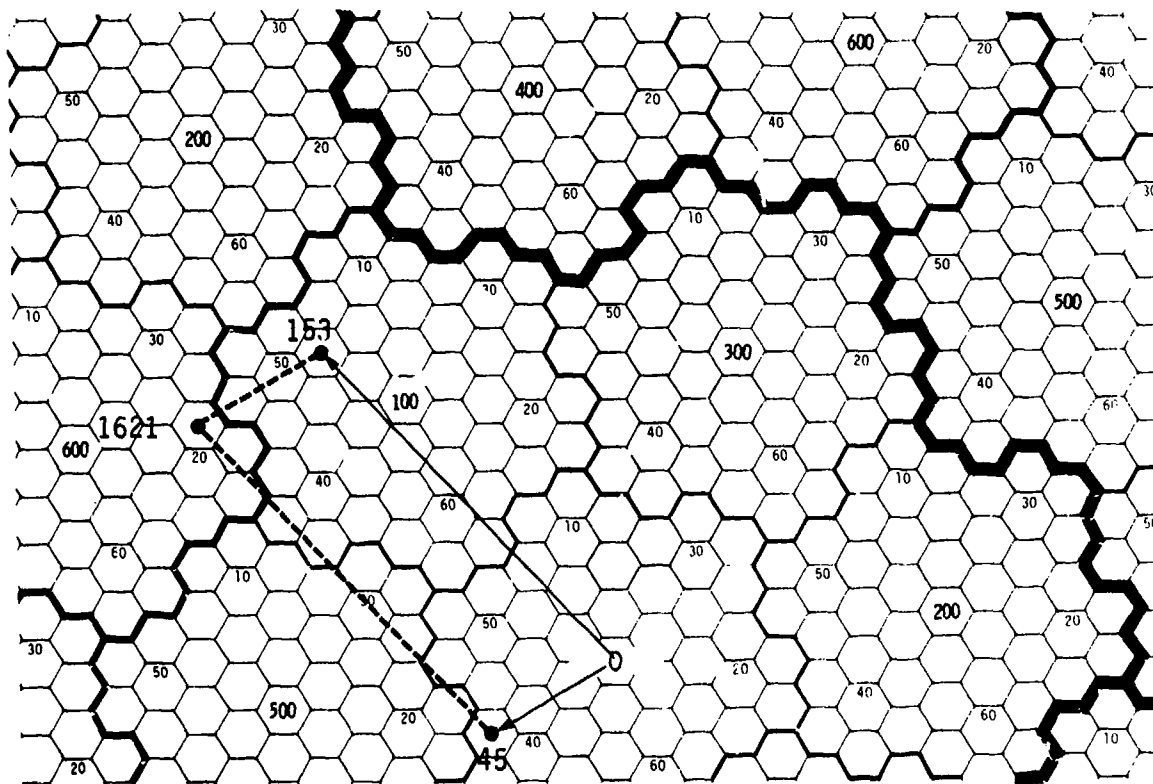


FIGURE 21: The GBT Sum of 153 and 45

•	0	1	2	3	4	5	6
0	0	0	0	0	0	0	0
1	0	1	2	3	4	5	6
2	0	2	4	6	1	3	5
3	0	3	6	2	5	1	4
4	0	4	1	5	2	6	3
5	0	5	3	1	6	4	2
6	0	6	5	4	3	2	1

TABLE 3: GBT Multiplication

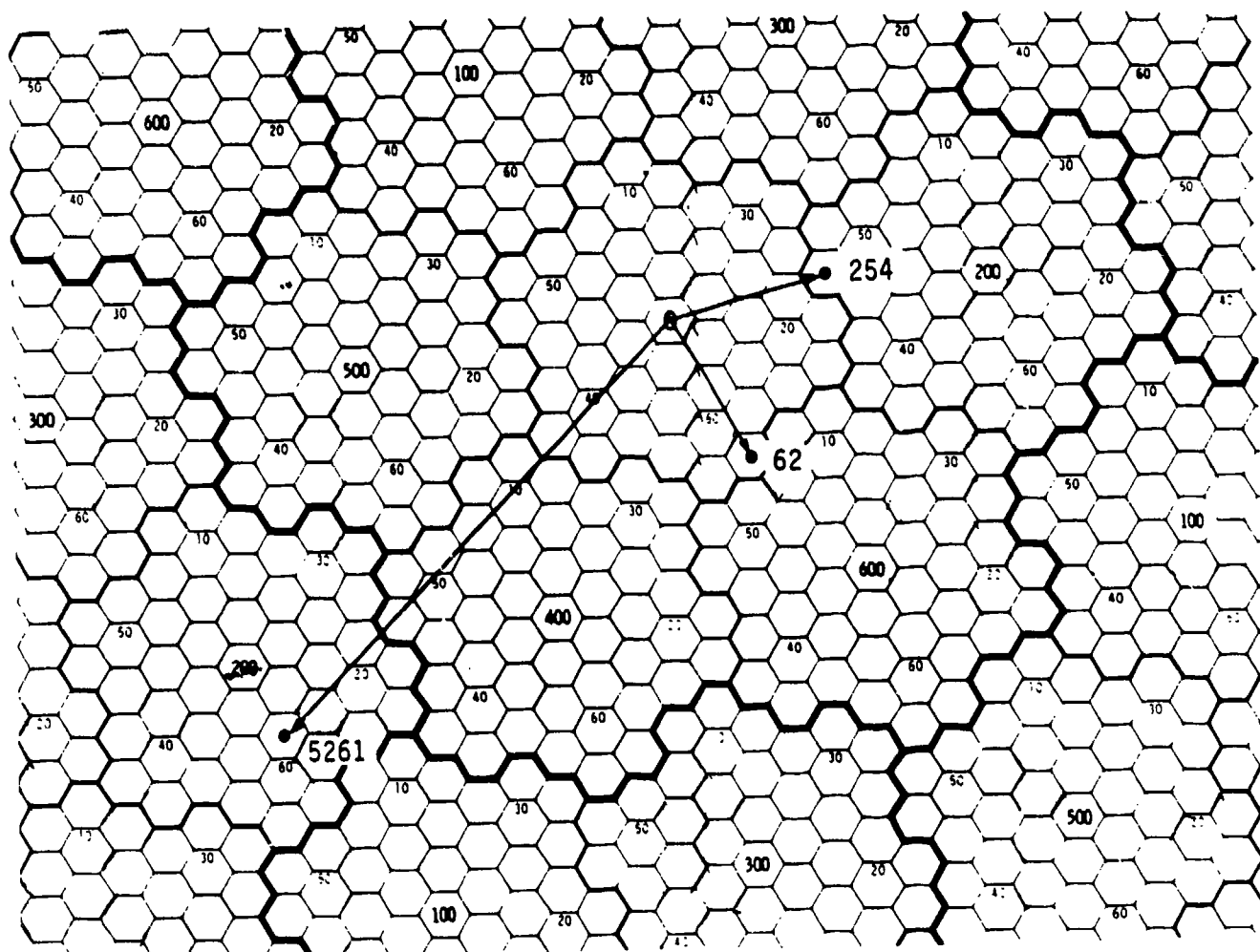


FIGURE 22: The GBT Product: 254 x 62

4.2.3 The GBT Data Base Structure

Any set of GBT addresses generates a tree structure. The tree derived from a list of 8 four digit GBT addresses is shown in Figure 23. This tree originates from a root node which corresponds to the universal cell containing all addressable cells. Attached to the root node is a node corresponding to the first of a set consisting of those next lower level cells beneath the universal cell which contain any of the cells on the list. Generally, each node in the tree corresponds to a cell in the GBT cellular hierarchy. Every node within a particular level of the tree structure is linked to its siblings in sorted order by the digit of the GBT address corresponding to the level. The first node within the list is attached to the parent cell for that level. Any given node can have, at most, seven subordinates, but may have fewer.

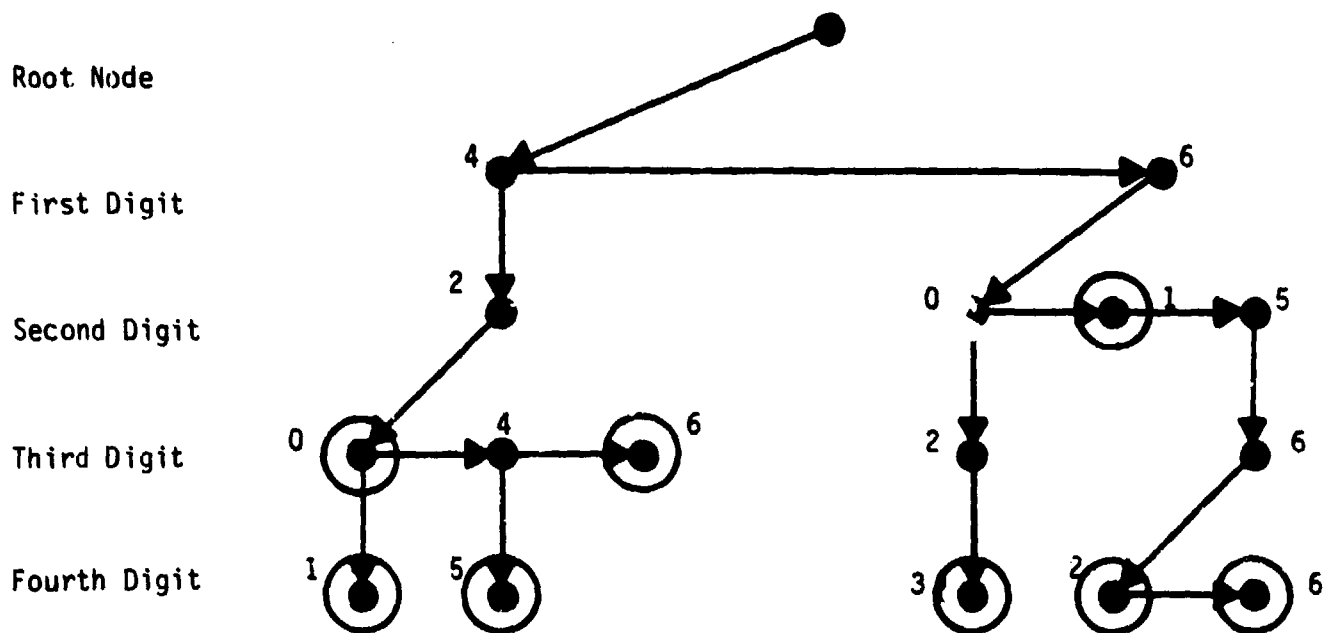
This tree structure provides an easy way of accessing data by general location. Figure 24 shows the location of the cells whose addresses are listed in Figure 23. Suppose someone were to ask for a list of the addresses stored in the tree which are within four units (1 unit is the distance between hexagon centers) of 4221. A simple computation shows that any such address must be a subordinate of one of the cells 4277, 6577, 6477, and 43177. The latter cell is not part of the four digit universe used in this example, and therefore, is not checked. To check the contents of 4277, we start at the tree root, follow a pointer to cell 477, then to cell 4277. Then we follow all pointers from the 4277 node to discover the addresses 4201, 4207, 4245, and 4267. These are placed on a list of candidate addresses. To check 6477 and 6577, we first descend from the root node to cell 4777 then to the node corresponding to 6777. At the cell 6777 information is stored within the cell describing the subordinates to the cell. It is at this point we discover that there is no pointer to 6477, so there can be no stored addresses subordinate to that cell. We then follow the pointer from cell 6777 to cell 6077 and from there to cell 6017 and finally to cell 6577. By following all subsequent pointers from cell 6577 we locate cells 6562 and 6566 which are added to the list of candidate addresses. The next step is to use the GBT subtraction operation and vector length measurement to eliminate those addresses on the list which are more than four units from 4221. This step eliminates 4245 and reduces the list to 4201, 4207, 4267, 6562, and 6566, all of which are (at least partially) within 4 units of 4221.

FIGURE 23:

A List of Addresses and the Corresponding Tree Structure.

Nodes in the tree corresponding to addresses on the list are circled.

ADDRESSES: 4201, 4207, 4245, 4267, 6023, 6177, 6562, 6566



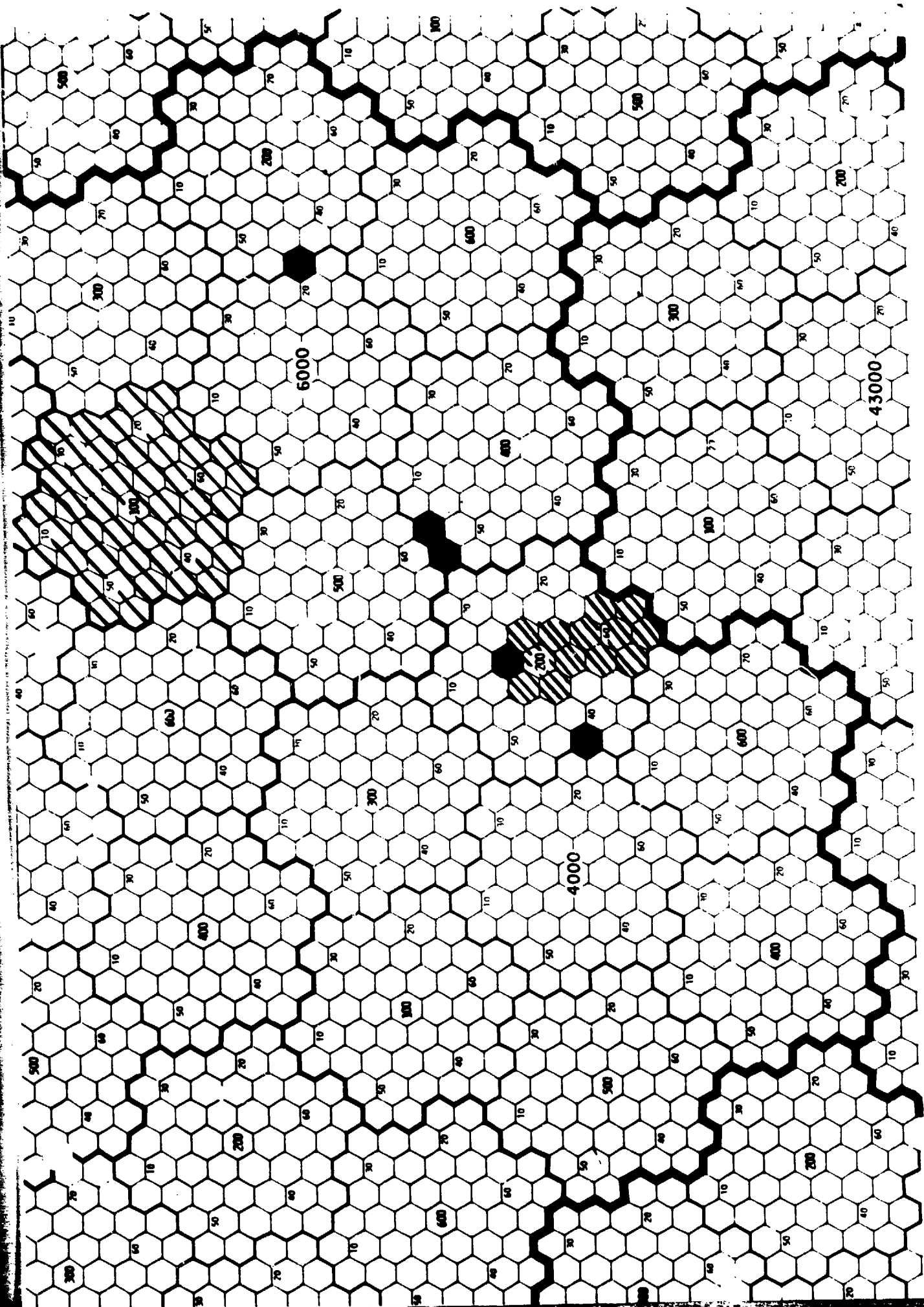


FIGURE 24: The Location of Cells on the Address List

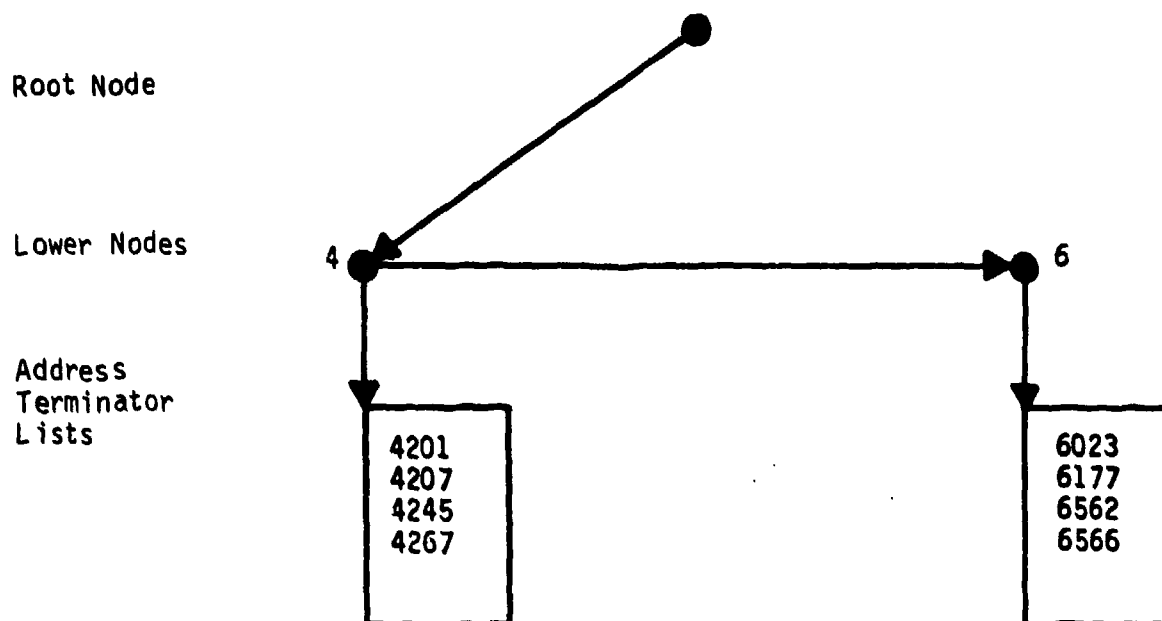
General GBT data accesses operate in the same manner as the above example. If a region R is to be searched, the addresses of a small number of large cells which cover R are determined. Then these cells are searched by examining their subordinate nodes in the data structure until all stored subordinates of these cells are accessed. Then, this list is checked against the boundary of region R to determine which cells actually intersect that region. This is a relatively direct procedure for accessing by location in that it avoids any consideration of addresses which lie outside the set of cells which cover R. This locational access technique is not of great utility if the number of addresses stored in the data structure is small. However, in large data bases, maps or imagery, for instance, which contain multiple millions of points, some such access mechanism is imperative to rapid operation.

The tree structure shown in Figure 23, and described above, is wasteful in that some nodes exist in the tree solely to provide a linkage between a larger cell and a very small number (1 or 2) of subordinates. These problems can be overcome by the structure indicated in Figure 25. This abbreviated tree structure does not contain separate nodes for many of the parent cells of addresses which are stored. Instead, the suffix digits of a stored address, or stored addresses, are listed in a node corresponding to an address which may be several levels above the stored addresses in the hierarchy. A new level in the abbreviated tree structure is created only when the number of cells below the parent cells becomes large enough that it is no longer efficient to use list processing methods for the cells. The effect of the abbreviated tree structure is that stored addresses may be accessed by passing through fewer nodes than in the full tree structure of Figure 23. The locational access properties of the full tree are preserved in the abbreviated version.

FIGURE 25:

An Abbreviated Tree Corresponding to the same Address List as Figure 23

The address terminator lists shown in boxes are pointed to by the node just above them.



4.2.4 Data Types and Representations

A two dimensional data structure must be capable of storing data which is zero, one, or two dimensional. Sometimes, these distinctions are not clear. On large scale maps, for instance, a road can be stored as the area bounded by the curbs or shoulders while on small scale maps that road can be treated as equivalent to its center line. For a fixed resolution level, however, it is reasonable to assume that any data type can be assigned a dimensionality. Taking maps again as an example, prominent peaks might be stored as points, roads and streams might be stored as lines, and political units, forests, and census tracts might be stored as areas.

In a GBT data structure all these data types are represented using tree nodes, bit maps or flags stored at the nodes, and attribute records linked to the nodes by pointers. A mountain peak which is stored as a point has a GBT address that corresponds to its geographical location. That address has a node associated with it in the GBT tree. The mountain peak could be represented in a GBT file by the presence at that node of a "mountain peak" flag and a pointer to a record which contains attributes of the peak such as its name and elevation. Line data, for example a road, is represented by a set of points. Again the points have GBT addresses and the line is represented in the file by the presence of their corresponding nodes. At each node a road flag is set. The connections between points are stored as GBT vectors in the attribute records. If point A is connected to points B and C then the node for A has a pointer to an attribute record which contains the vectors B and C. Area data is represented by its boundary with attribute data stored in the attribute record of its (approximate) centroid. If a road is part of the boundary of an area, for example for a county, the points defining the road-boundary would have both a road and a county flag set. Figure 26 indicates the file and pointer structure involved in representing point, line, and area data.

In raster to vector conversion the GBT file structure is used to store easily computed descriptions of the pixel patterns in each aggregate. The algorithms can then work with these descriptors instead of the mass of pixel data.

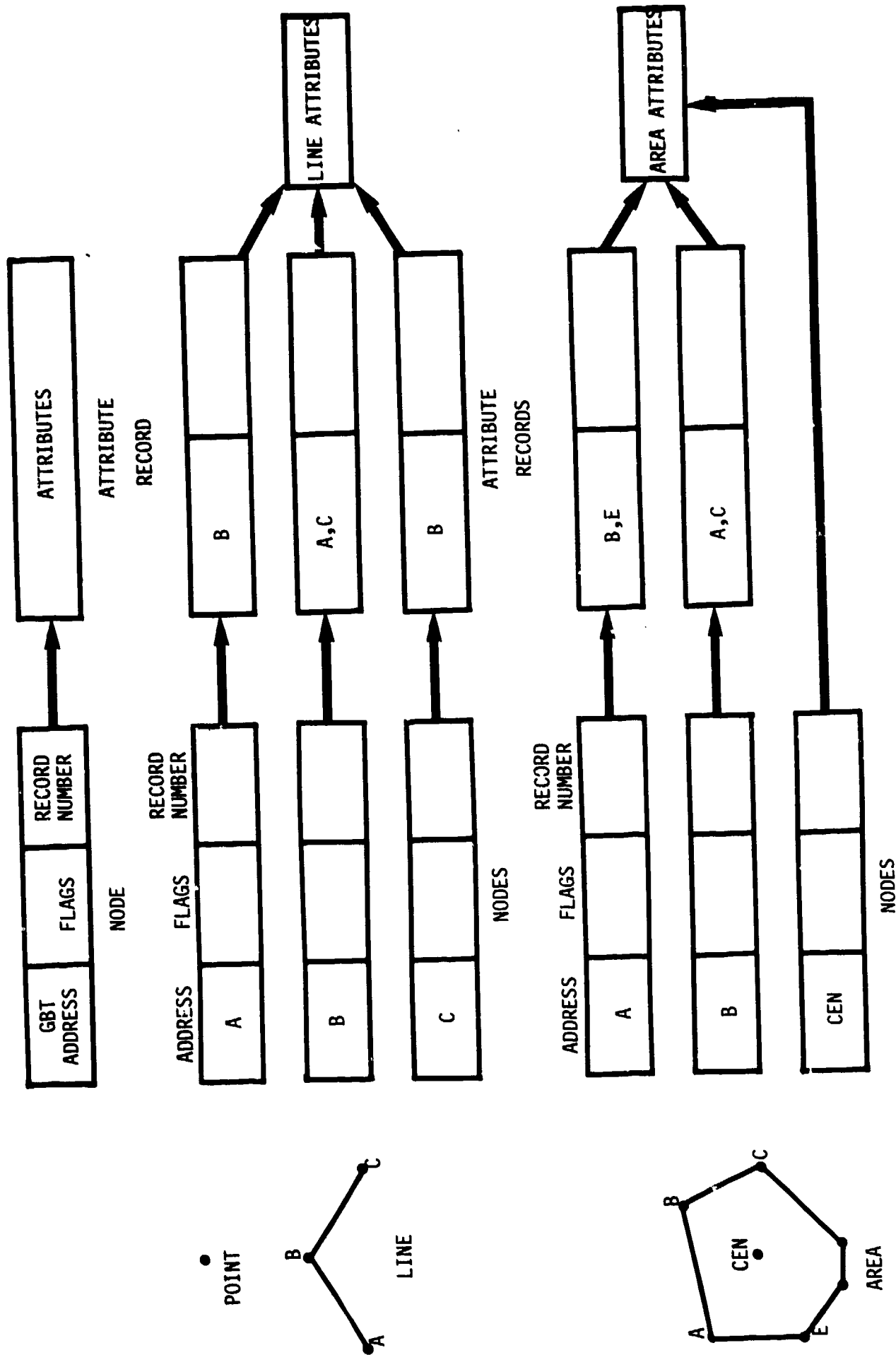


FIGURE 26: Data Types and their Associated GBT File Representations

5.0 Implementation

During the course of this project scanners and scanner technology were investigated. It was important to understand the quality of the data that could be obtained from a scanner in order to design the RVC algorithms. Since no scanner was available for most of the project, whenever possible, sample scans of maps were obtained from manufacturers. These samples were used to refine the algorithms. In the last stage of the research, ISC purchased an Optronics 10 inch by 10 inch rotating drum optical scanner. Such factors as quality of output, cost, size, and ease of computer interface were considered in making the choice. The test data sets described in this section were generated with this scanner.

Most scanner systems share certain characteristics. A light source shines on or through the scan medium (e.g., paper, mylar, film). A sensor detects how much light is reflected or passes through and produces an analog signal. This signal goes through an analog to digital converter. Either the light source or the paper is then moved and another portion of the medium is sampled.

In the Optronics scanner (Colorscan System C-4100), the light source is a tungsten-halogen lamp. The medium spins on a drum with each revolution producing one scan line and the illuminator and detector move along the axis of the drum one unit for each scan line. The resolution of the scan can be varied from 12 to 400 microns in preselected increments. The scanner sends its output to the VAX 11/780 through a 16 bit parallel interface. There the output is run-length encoded and put in a file on the disk. This process is able to keep up with the scanner.

The software system implemented during this research was designed to take such a run-length encoded scan file as input. At first these files were generated analytically and later taken from tapes of scans made elsewhere.

The experimental software was built as a tool for algorithm development. It makes use of computer graphics to study in detail the working of the algorithms. An RVC data base which contains the raster data, aggregate descriptors, and the final vectorized information can be accessed through the display. Many of the figures in this report are hard copies of Megatek displays (e.g. Figure 2 through 14).

Figure 27 shows the structure of the system. The first step in the RVC process consists of taking the raster scan data or a simulated scan file and converting the set pixels to GBT addresses. These are put into a GBT file using GRAM, the GBT Record Access Manager. Data base population routines compute aggregate descriptors and put them into this (RVC) file. For developmental purposes a special display file is built. This file allows rapid display generation. The raster to vector conversion routines work with the RVC file, adding the vector data to it. The smoothing program is run on the RVC file, adding new

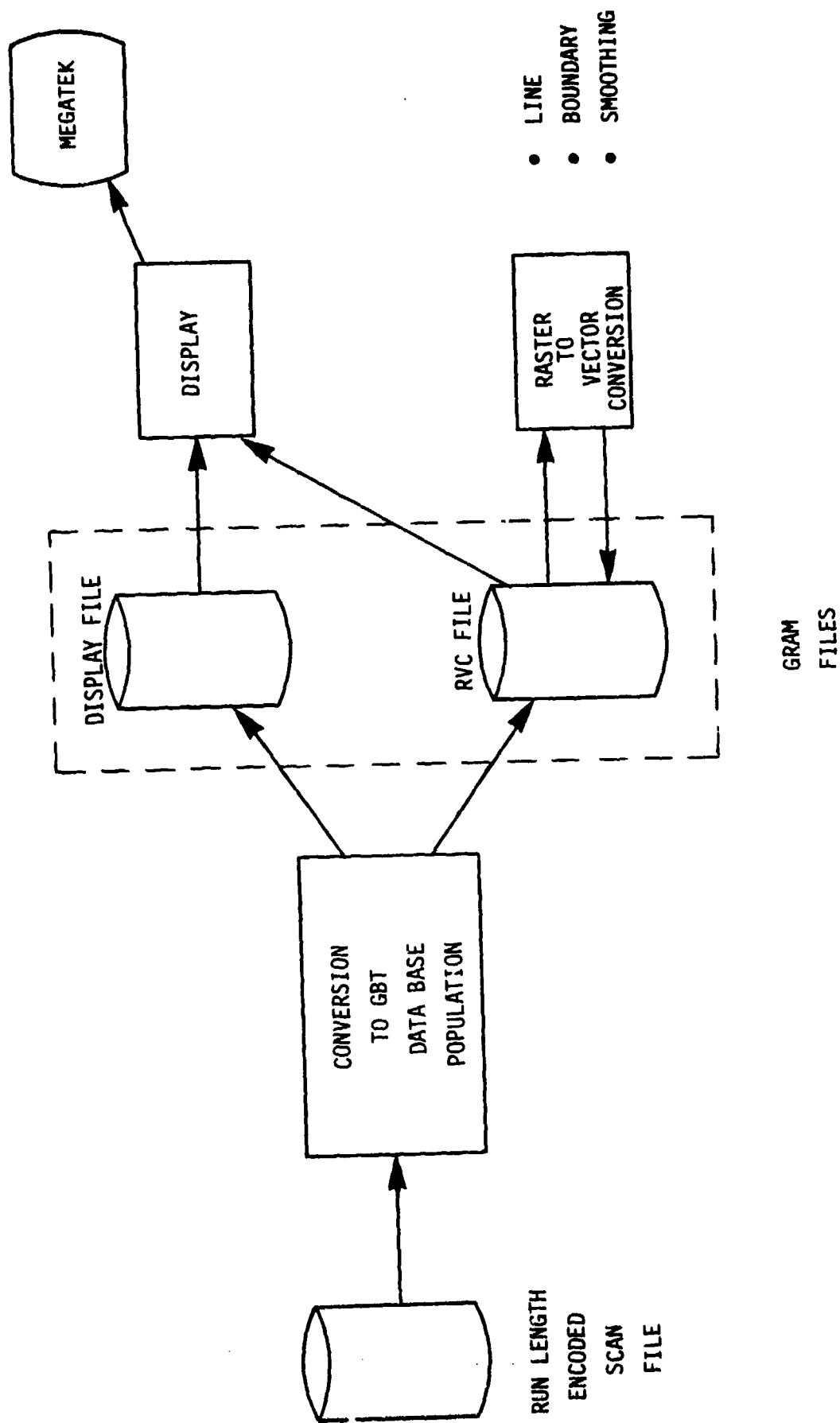


FIGURE 27: The Experimental Raster to Vector Conversion System

links and deleting old centroids according to the algorithm. Display routines use an interactive, menu-driven access scheme to build displays from the display file and, for the vector data, from the RVC file.

5.1 Test Data Sets

The experimental system was tested at the end of the project. Four different data sets were selected as representative of the data to be handled in four potential applications. The material scanned was:

- (1) A black line on mylar United States Geological Survey 7.5 minute series contour overlay for a portion of New Mexico. This was scanned at 25 microns.
- (2) A black line on mylar overlay of streets and buildings for the same area as (1). This was scanned at 25 microns.
- (3) A black line on mylar circuit diagram for a submarine. This was scanned at 100 microns.
- (4) An ink on white paper city planning map of Philadelphia. This was scanned at 50 microns.

Figures 28, 29, 30, and 31 show portions of the material that was scanned.

Figures 32 through 35 show details of the vectorization of the contour map (Data Set #1); Figures 36 through 40 show portions of Data Set #2 vectorized. Figures 41 through 43 are for Data Set #3 and Figures 44 through 49 show vectorizations of the Philadelphia map, Data Set #4.



FIGURE 28: A Portion of the USGS 7.5 Minute Contour Overlay used to Test the RVC System (DATA SET #1)

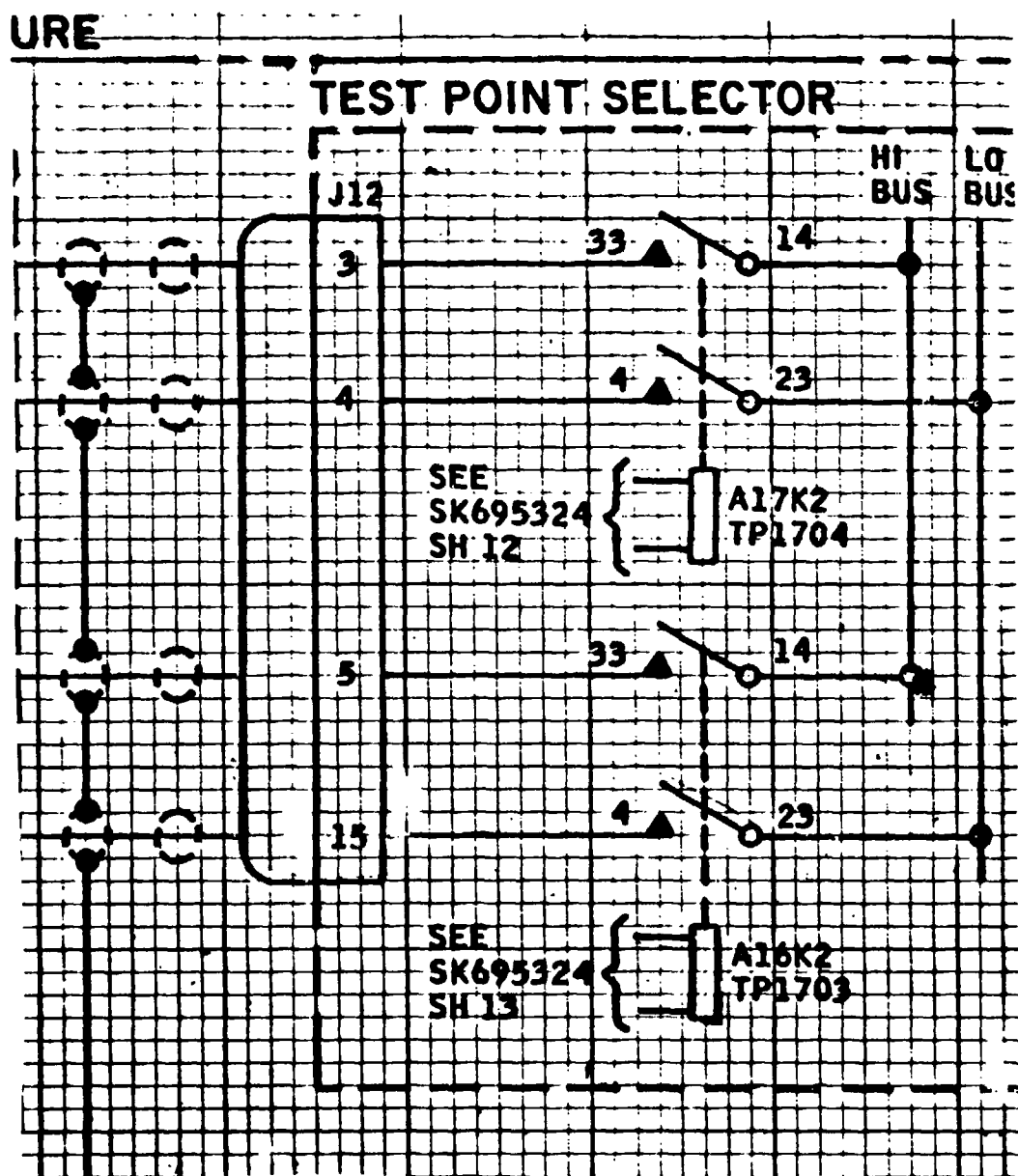


FIGURE 30: A Portion of a Circuit Diagram for a Submarine (DATA SET #3)

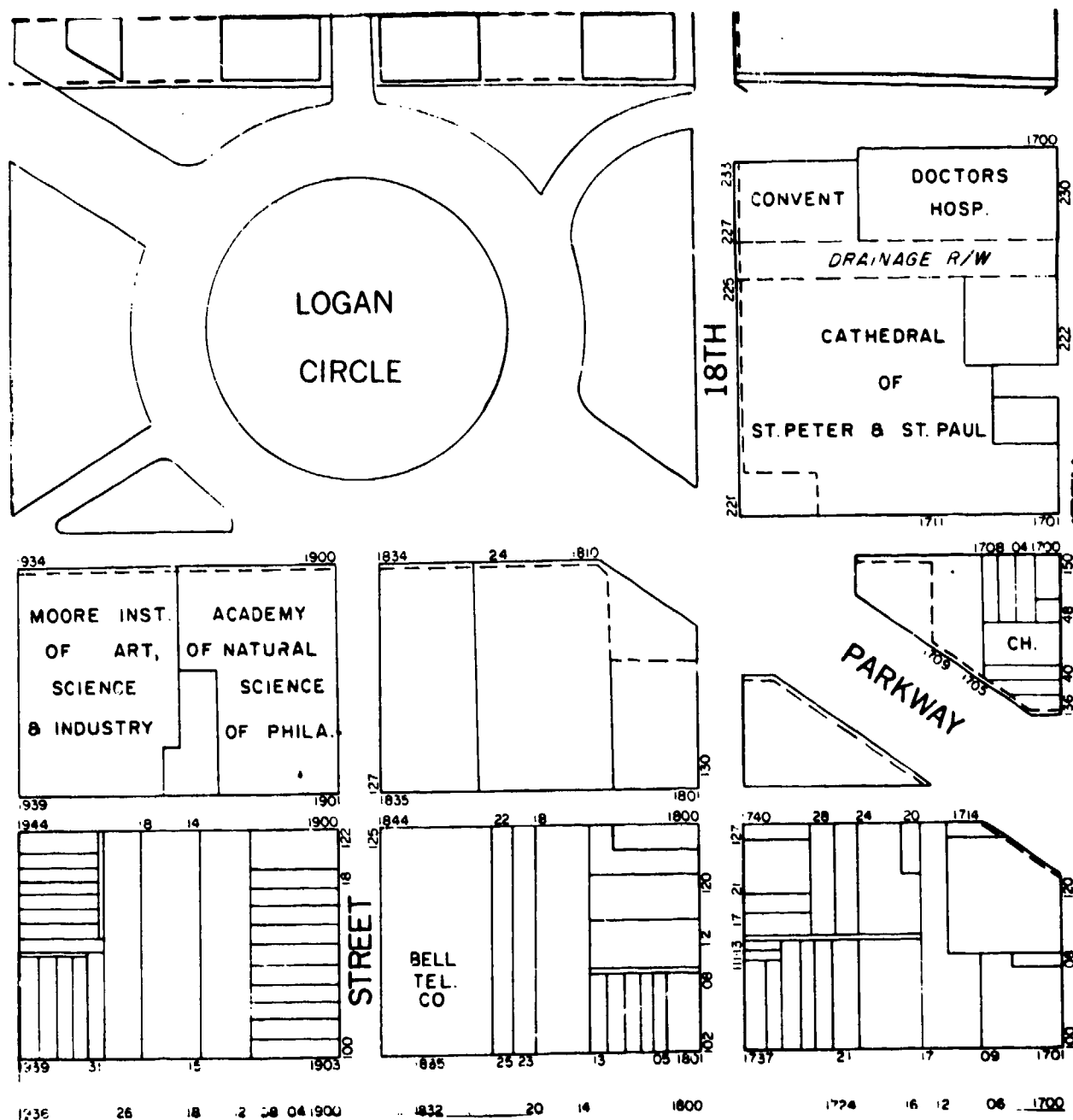


FIGURE 31: A Portion of a City Planning Map (Philadelphia) (DATA SET #4)

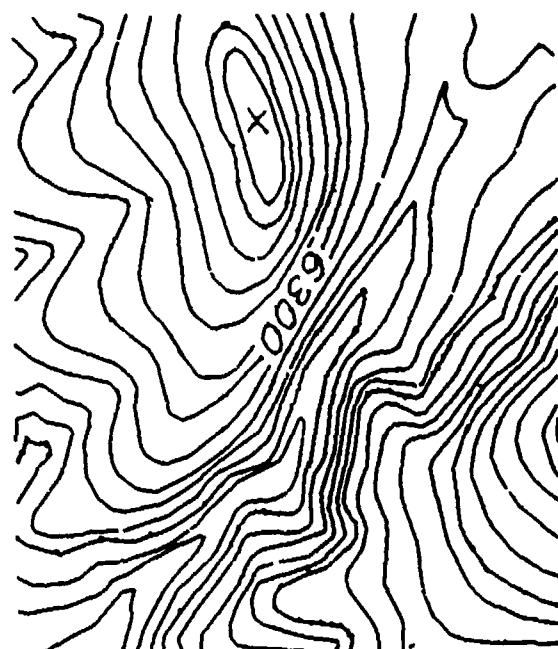


FIGURE 32: Vectorization of a 25 Micron Scan (DATA SET #1)

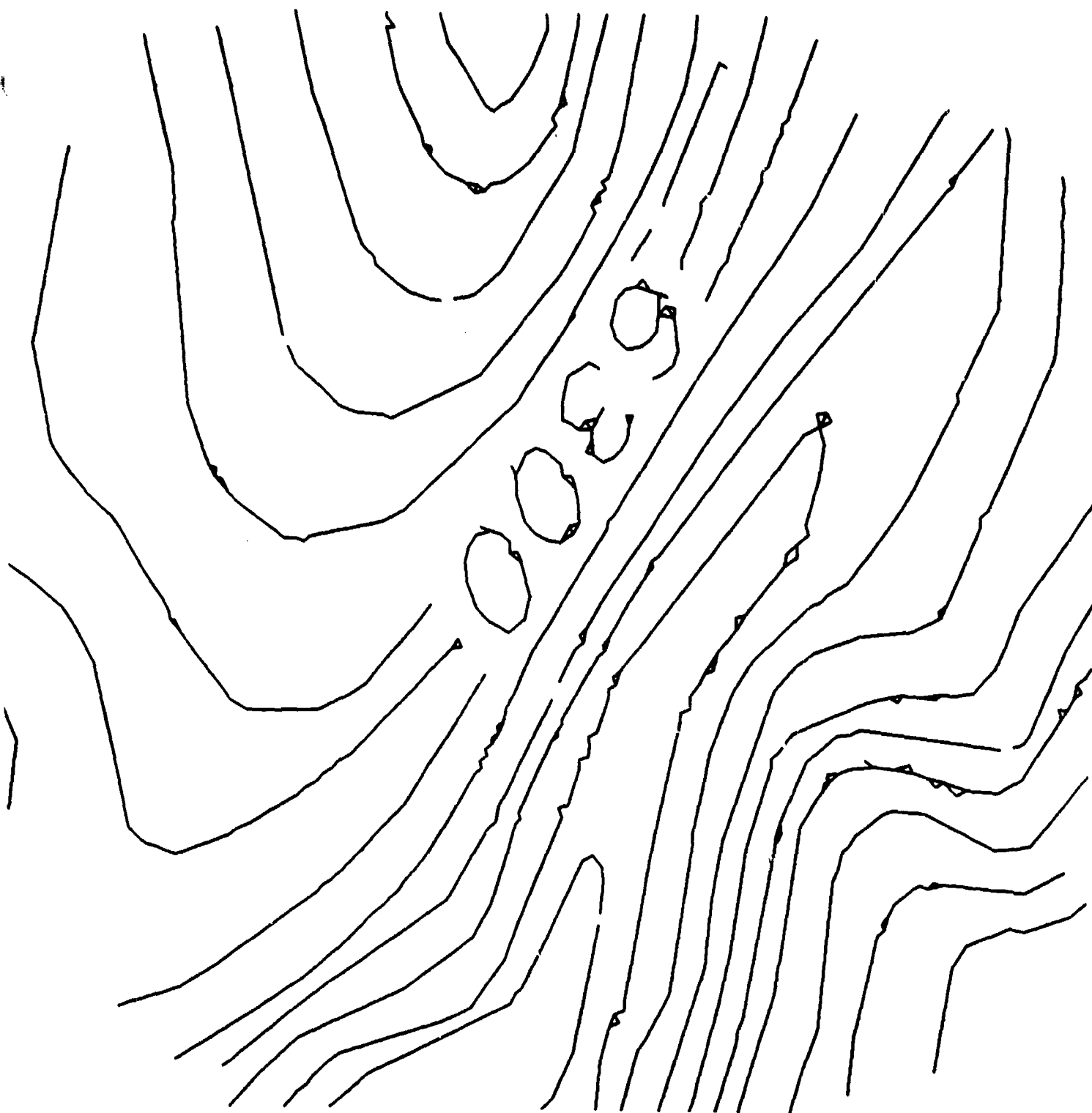


FIGURE 33: Close-up of Figure 32

Triangles and jogs in the line are due to improper handling of full pattern types.

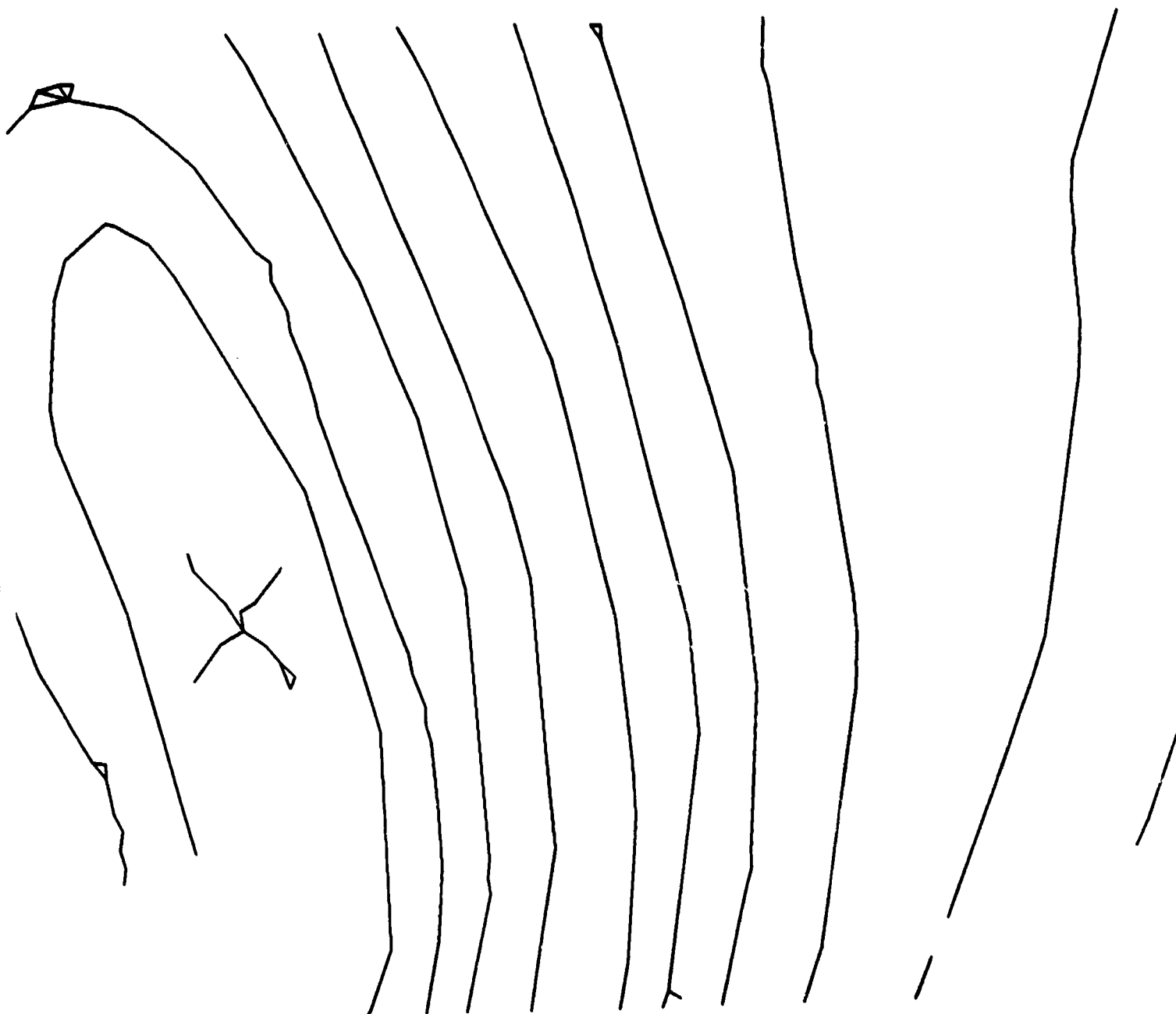


FIGURE 34: Close-up of Figure 32

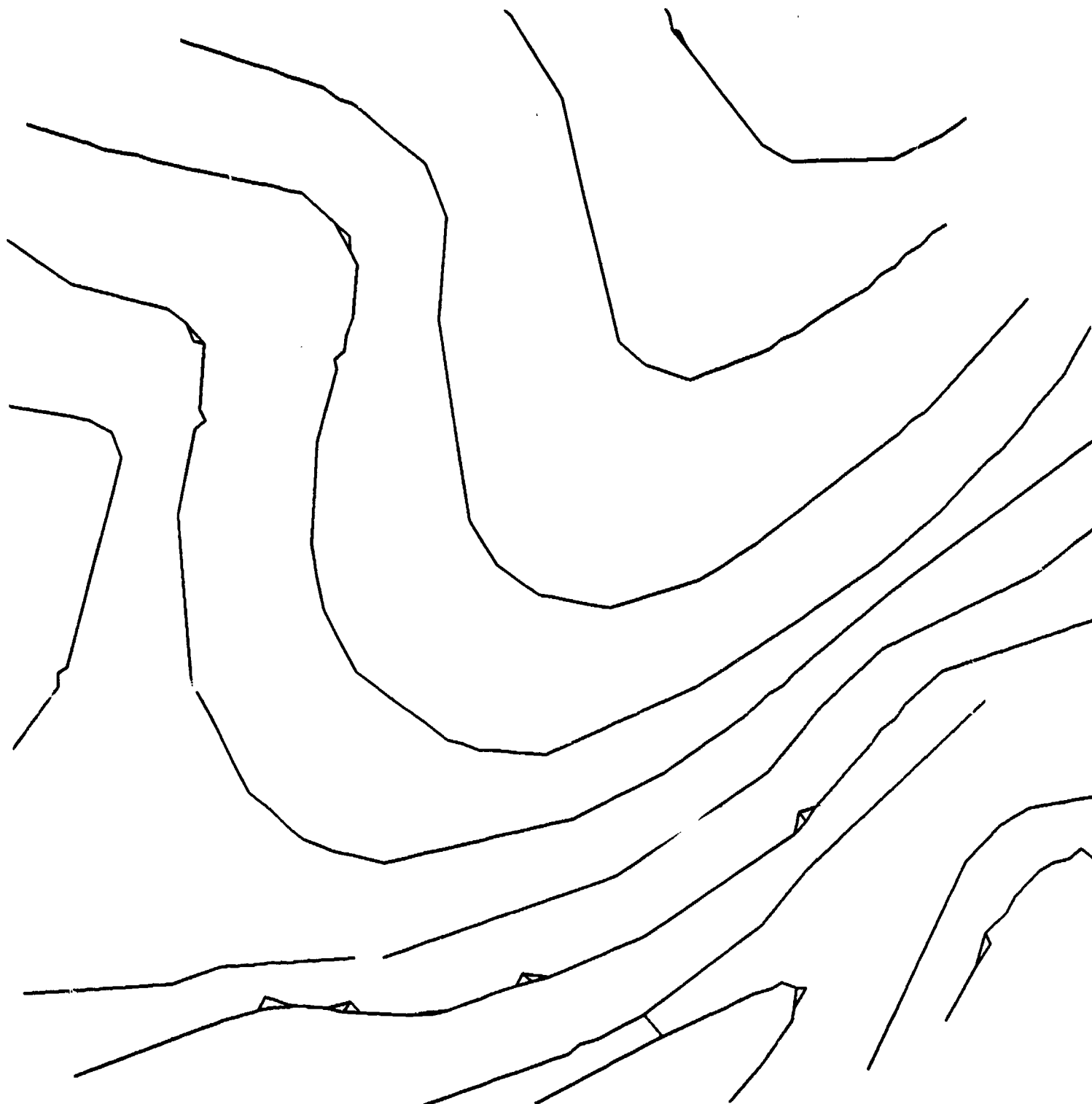


FIGURE 35: Close-up of Figure 32

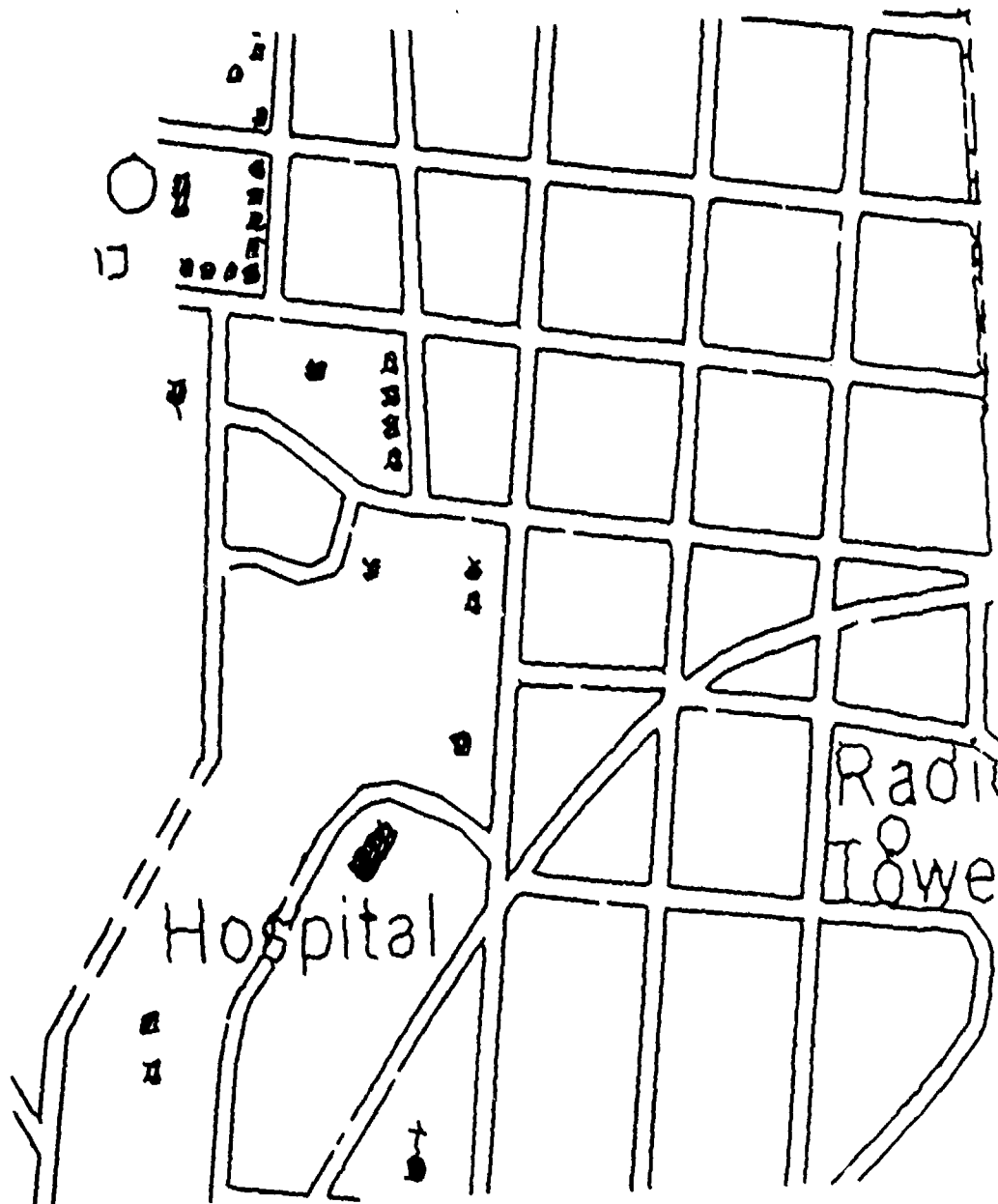


FIGURE 36: Vectorization of a 25 Micron Scan (DATA SET #2)

avel
Pit

X
= =
X

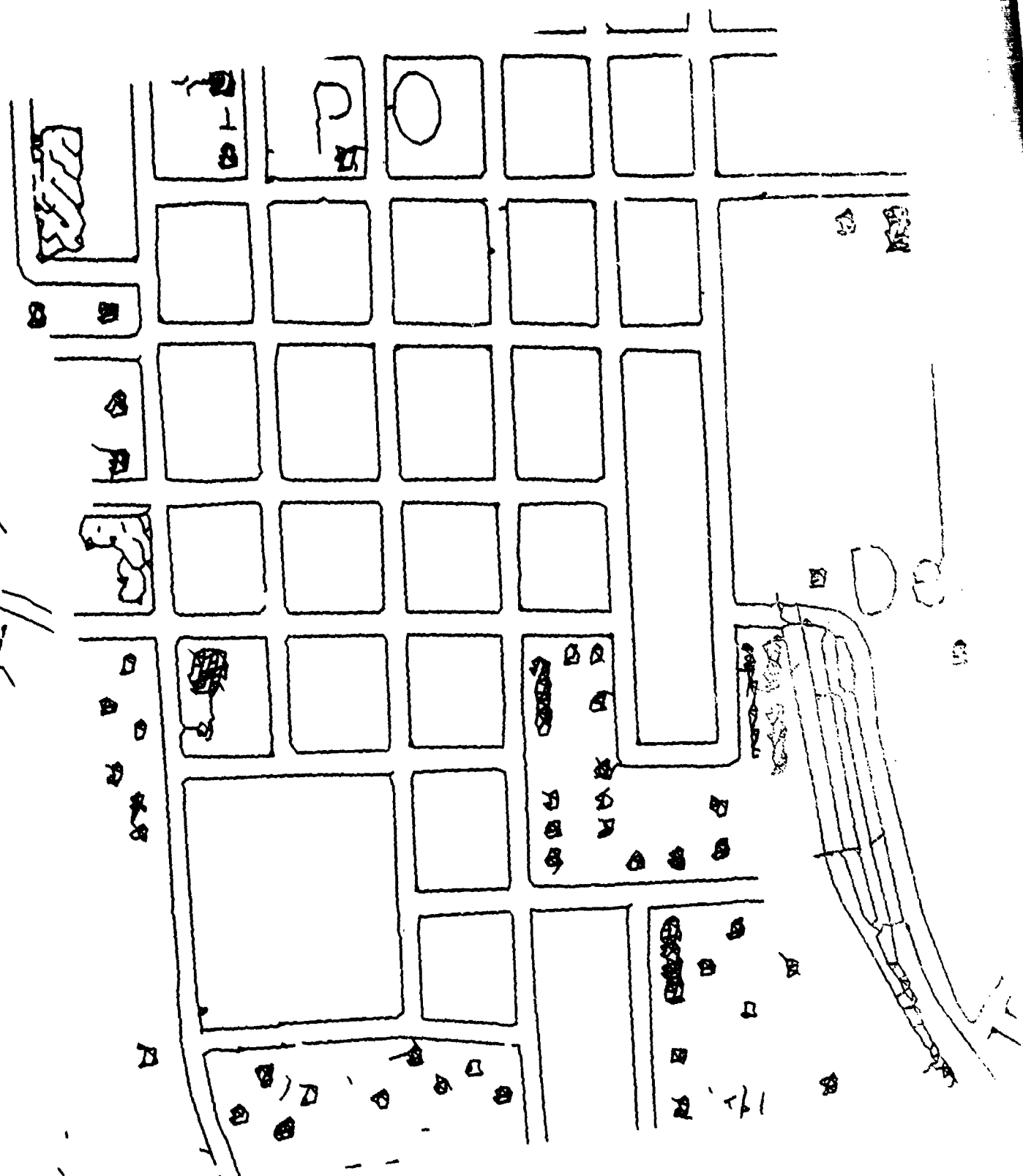


FIGURE 37: Vectorization of a 25 Micron Scan (DATA SET #2)



FIGURE 38: Close-up of Figure 36

Solid building symbols are not properly vectorized.

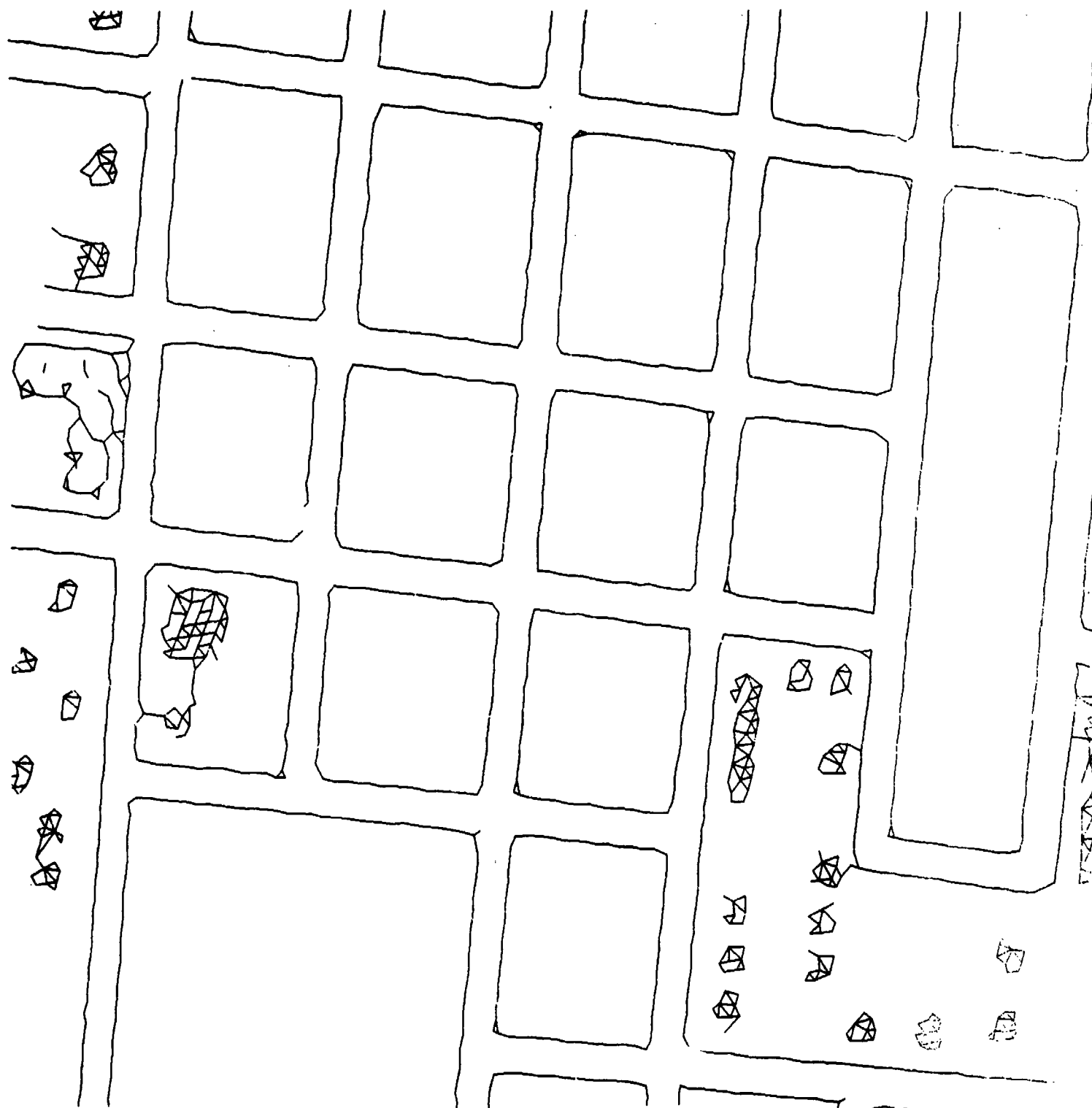


FIGURE 39: Close-up of Figure 37, unsmoothed

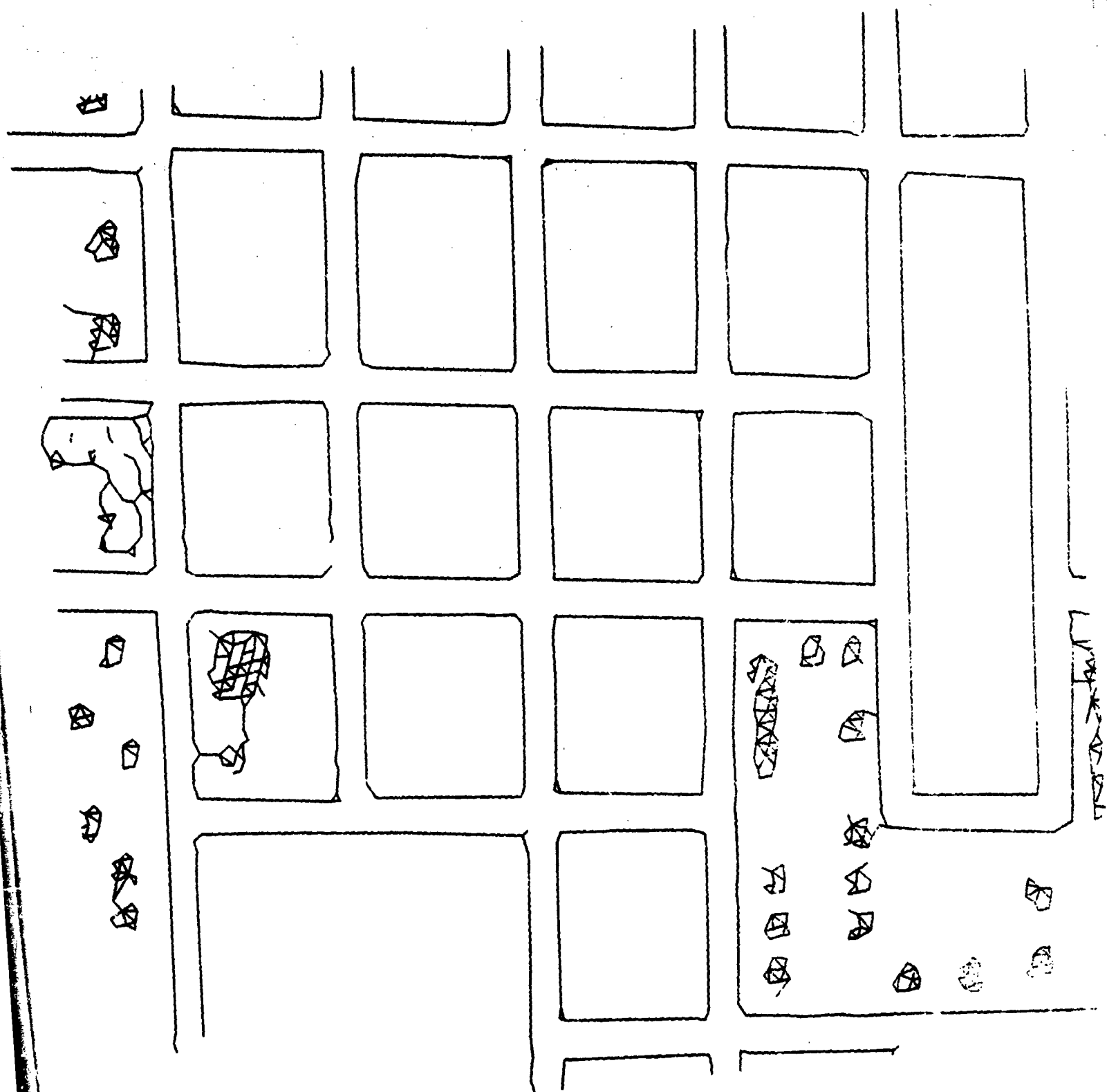


FIGURE 40: Same Area as in Figure 39, After Smoothing

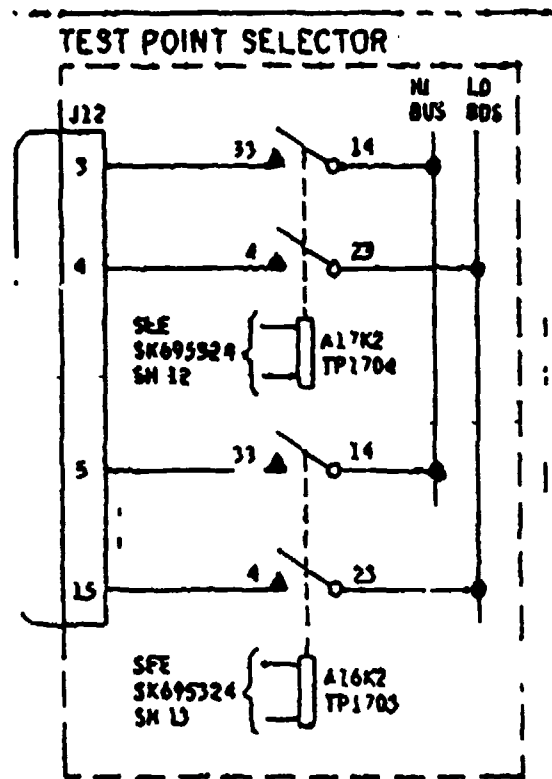


FIGURE 41: Vectorization of a 100 Micron Scan (DATA SET #3)

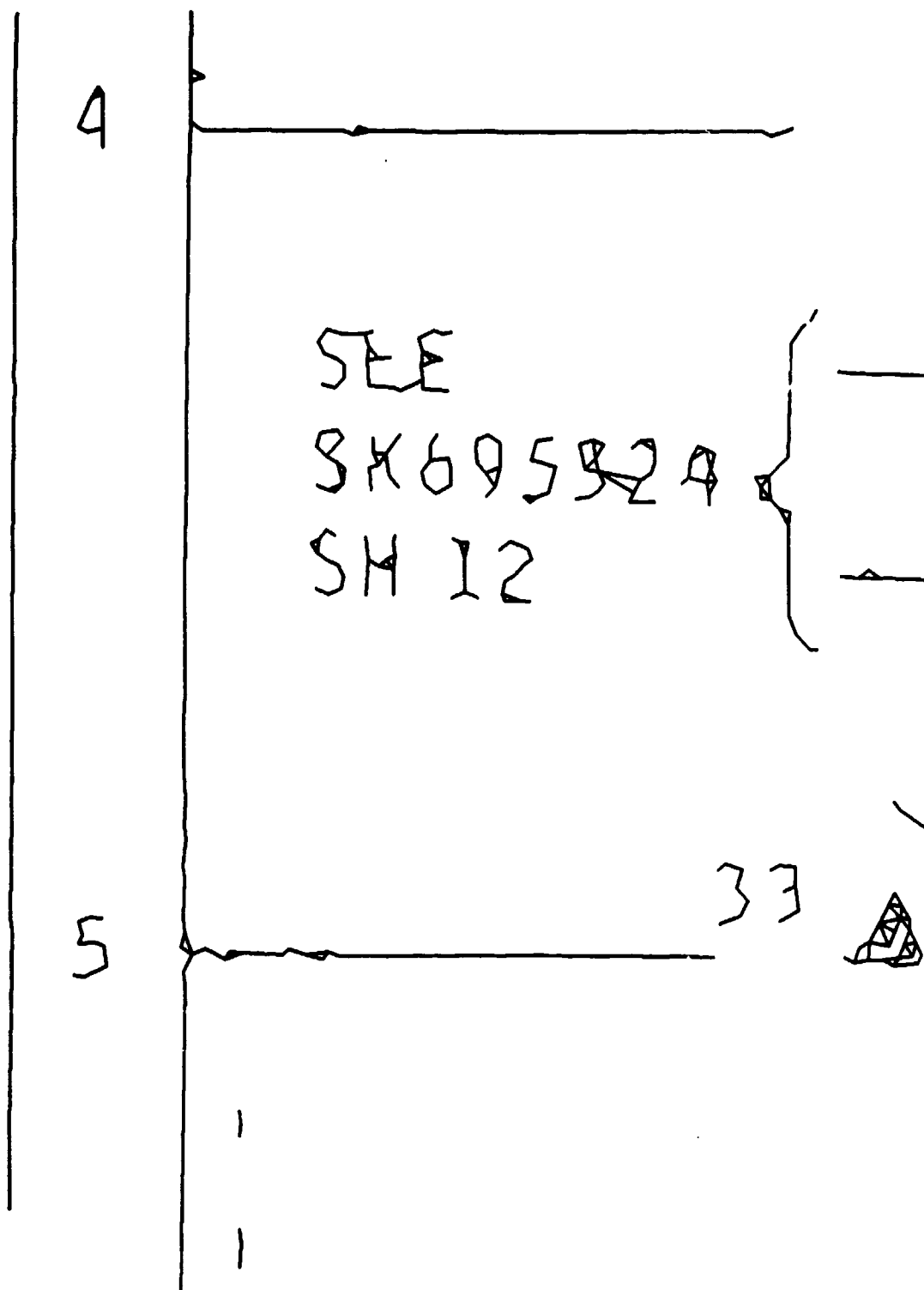


FIGURE 42: Close-up of Figure 41

Alphanumerics have extra links due to improper handling of pathological pattern types.

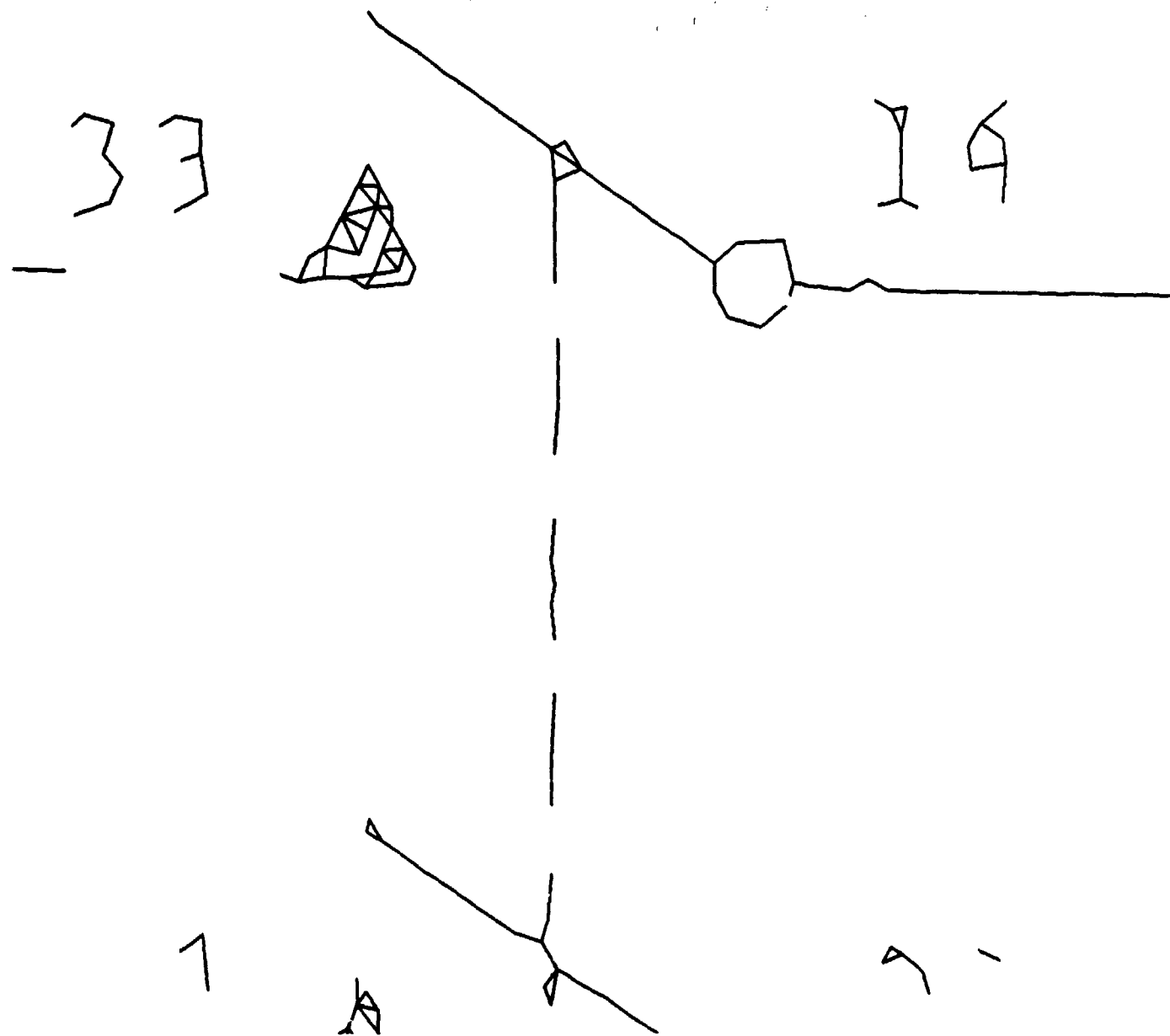


FIGURE 43: Close-up of Figure 41

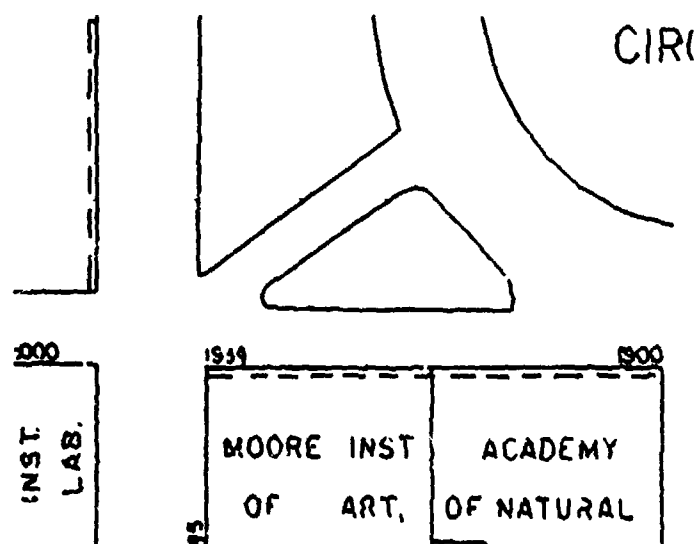


FIGURE 44: Vectorization of a 50 Micron Scan (DATA SET #4)

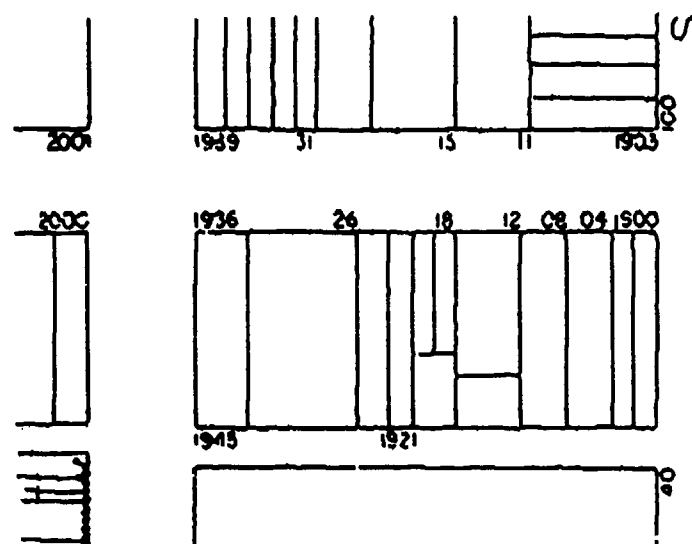


FIGURE 45: Vectorization of a 50 Micron Scan (DATA SET #4)

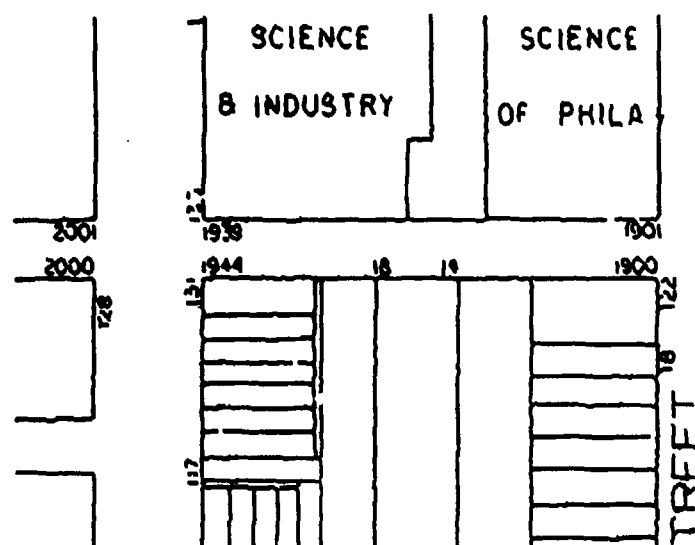


FIGURE 46: Vectorization of a 50 Micron Scan (DATA SET #4)

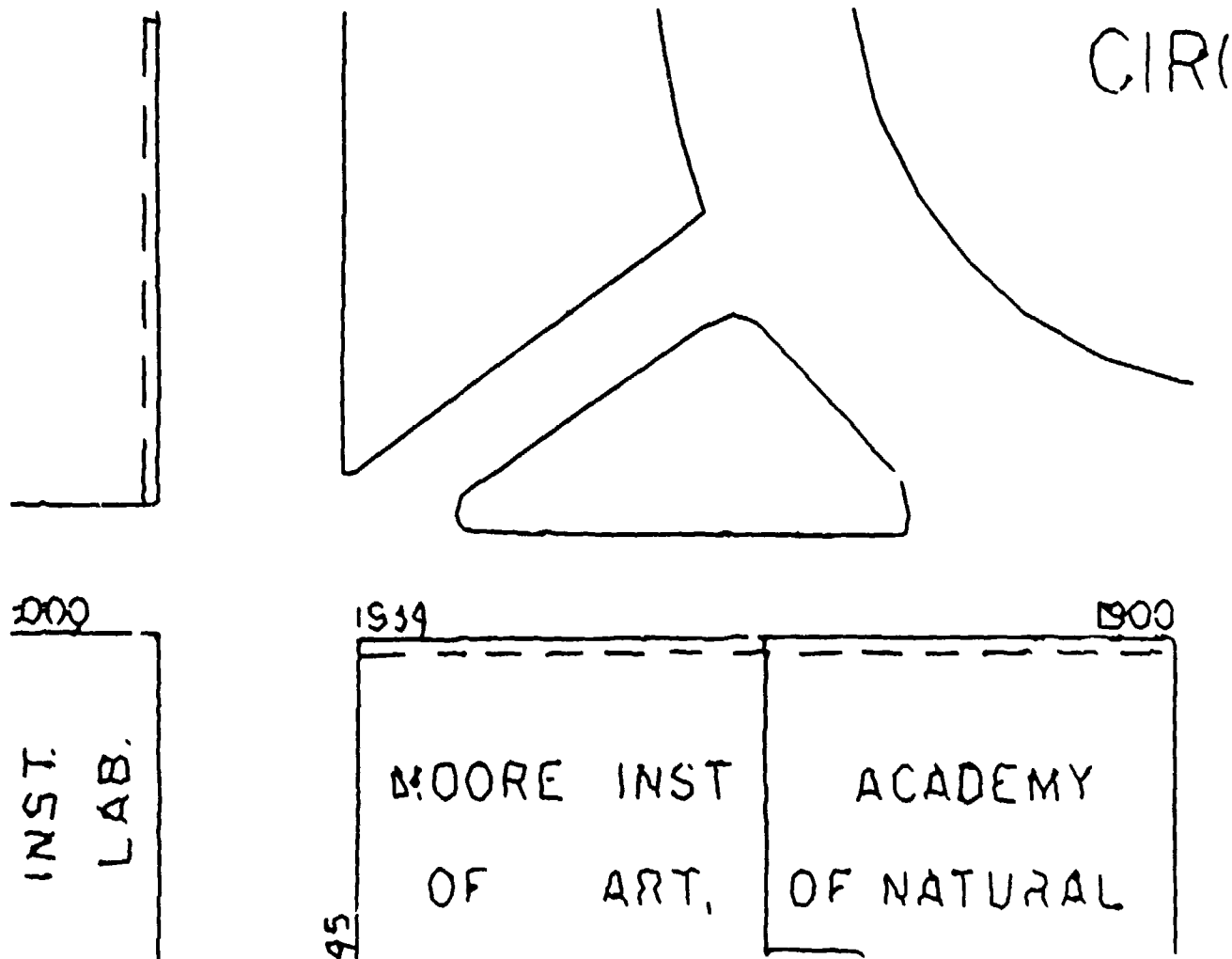


FIGURE 47: Close-up of Figure 44

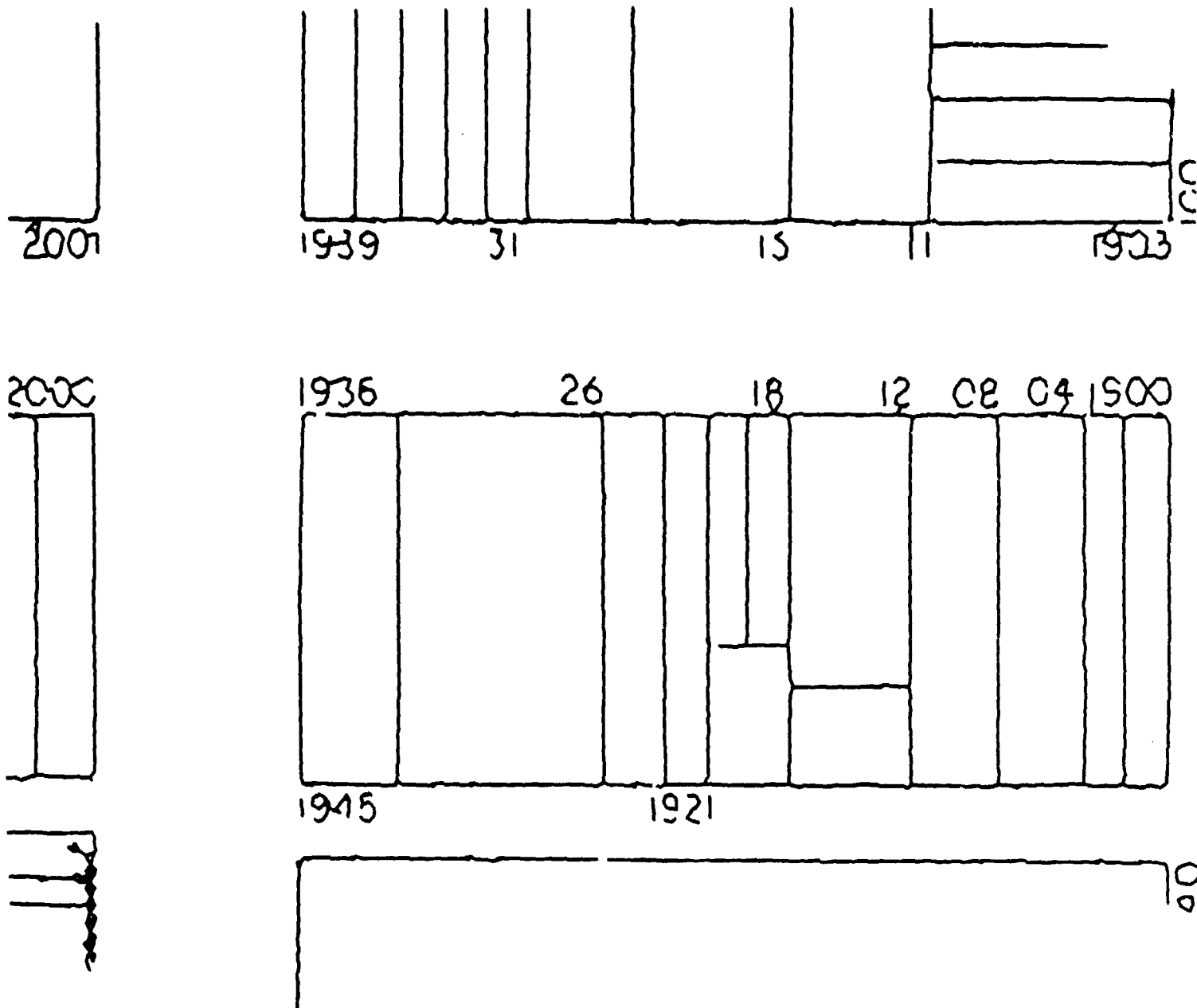


FIGURE 48: Close-up of Figure 45

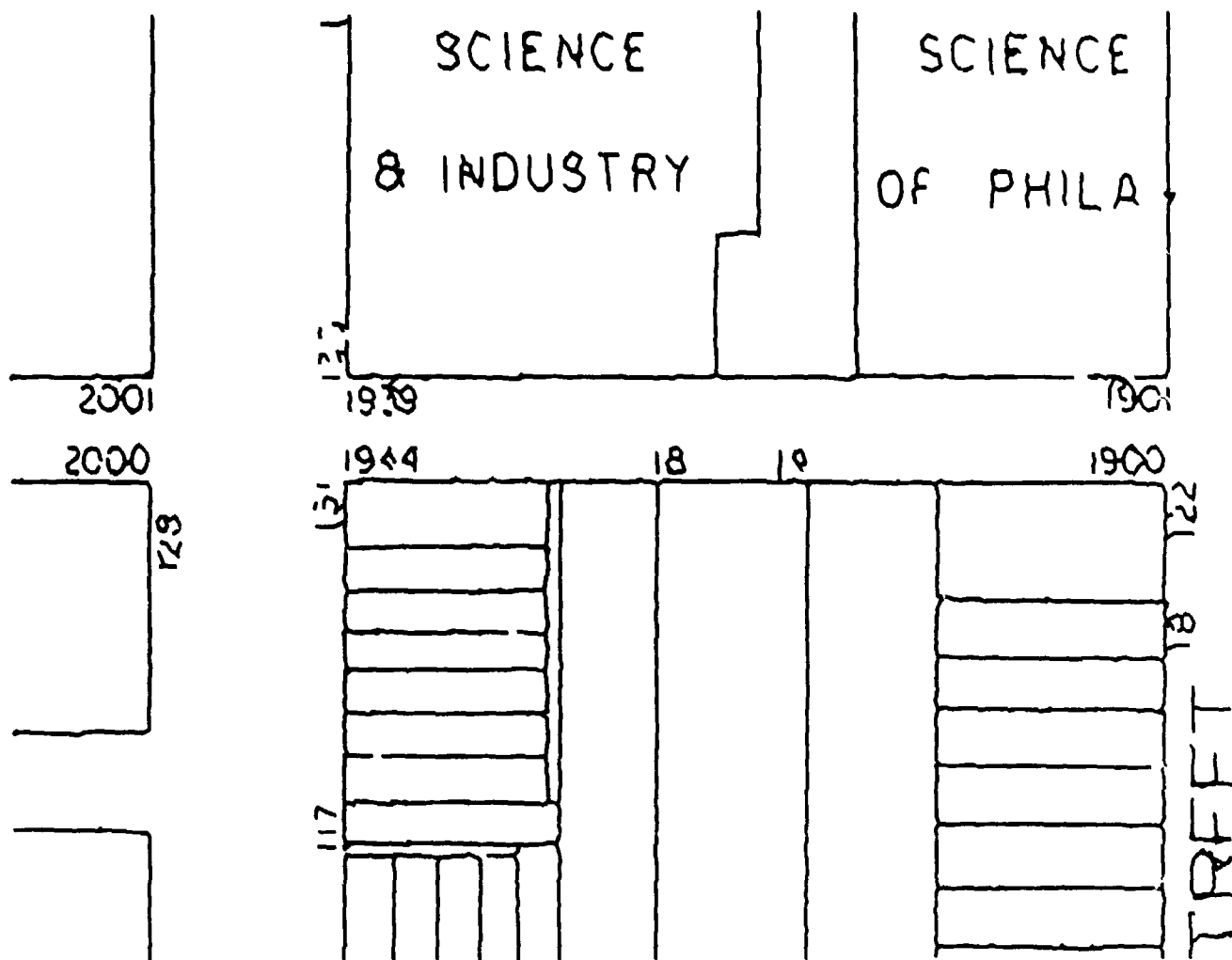


FIGURE 49: Close-up of Figure 46

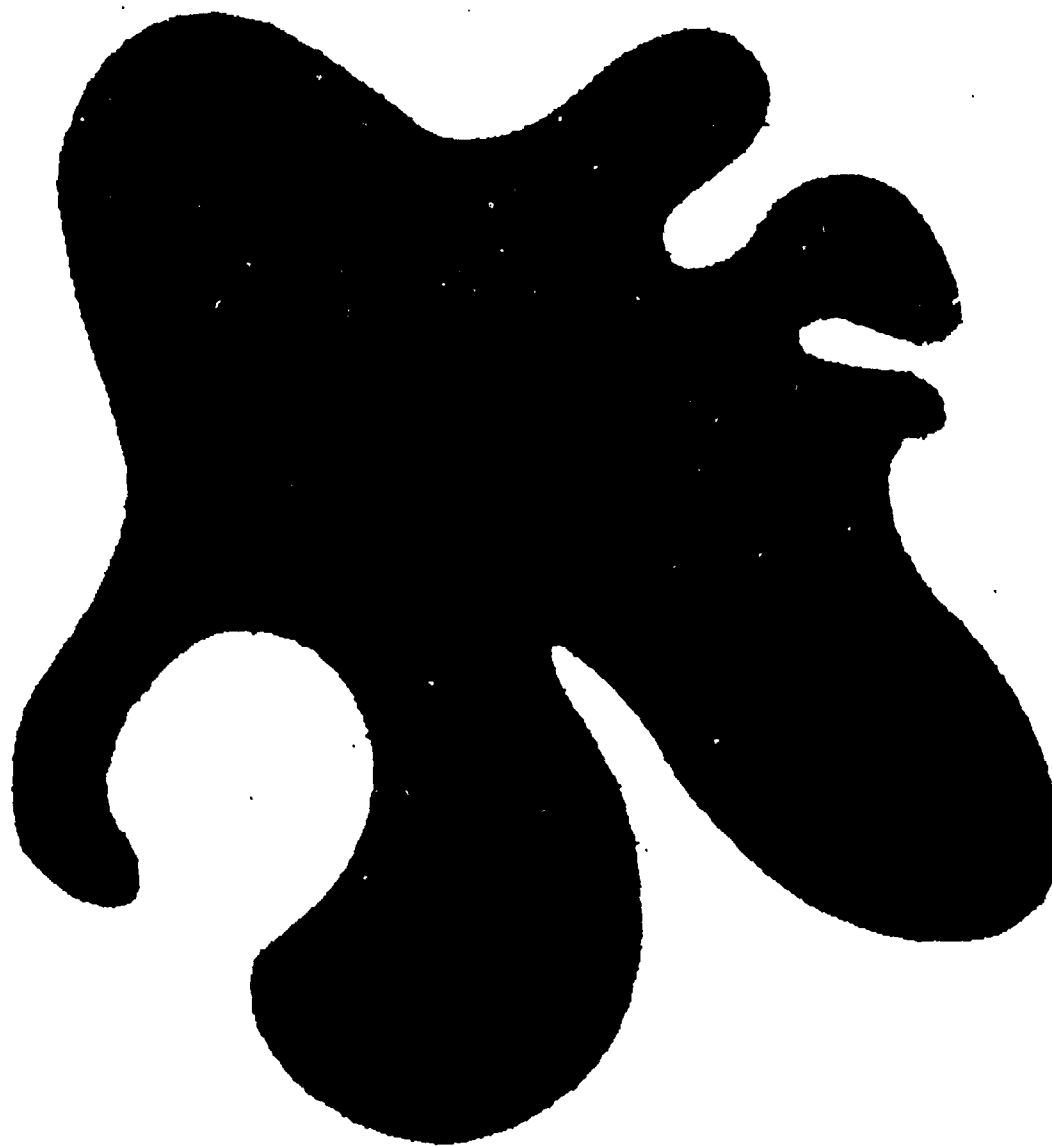


FIGURE 50: Ink Drawing Scanned for Boundary-Following

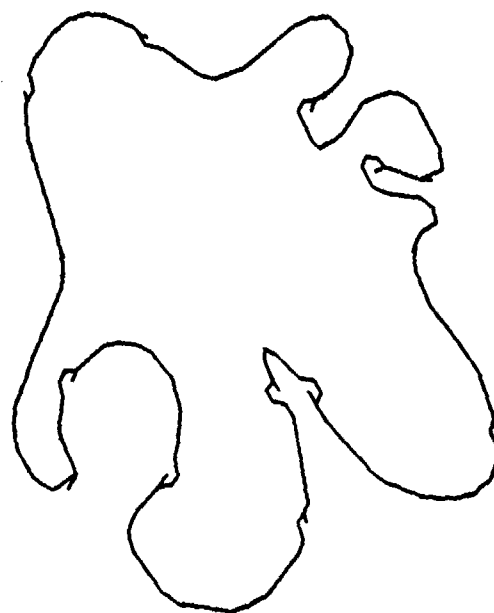


FIGURE 51: Vectorization of a 25 Micron Scan

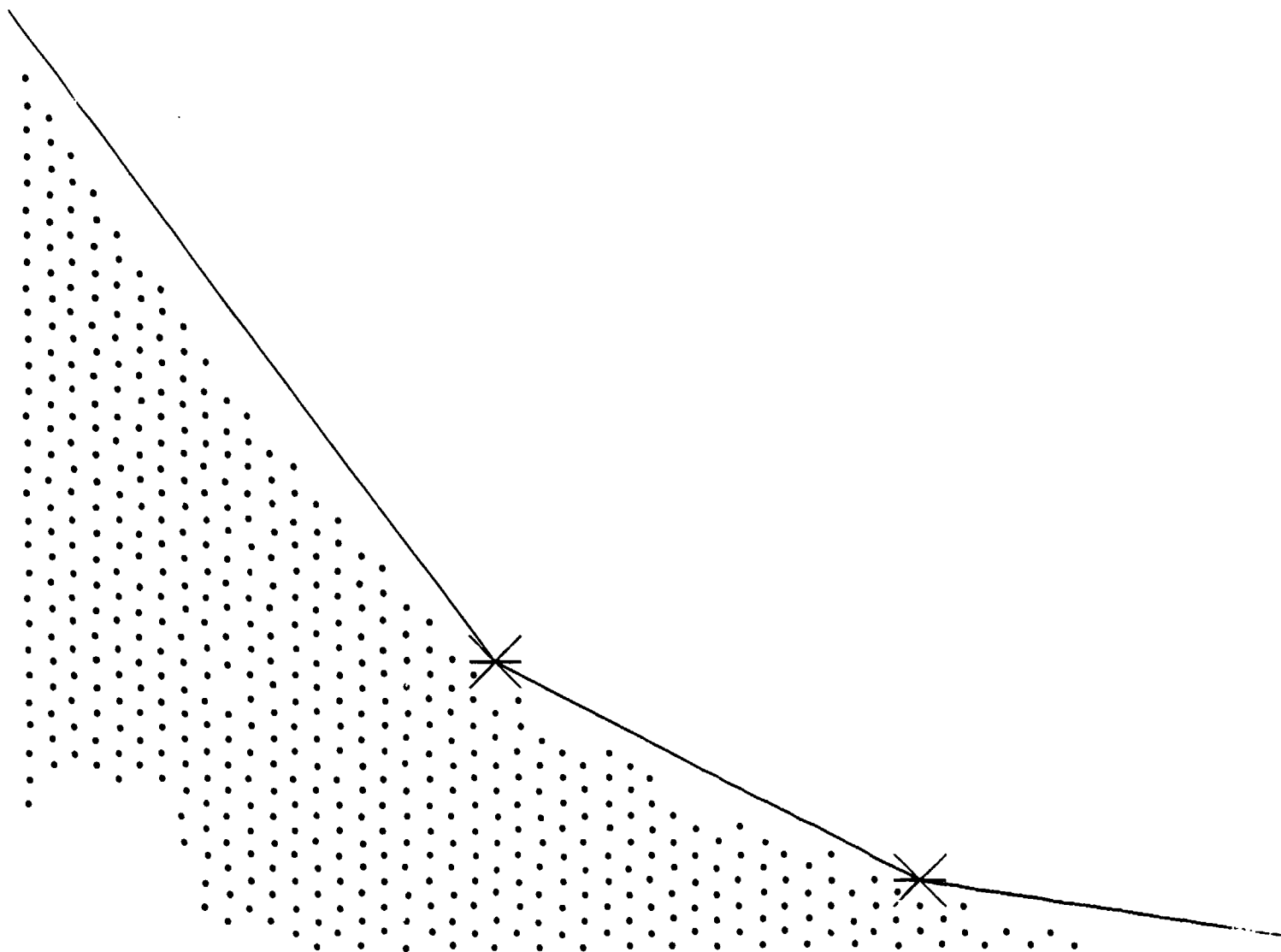


FIGURE 52: Close-up of the Vectorization of Figure 50

5.2 Conclusions

Most of the original notions of aggregate descriptors: gradients, weights, pattern types, and how they can be used by a line-following algorithm were unchanged by the results of this research. In fact, they were strengthened by examination of real scanned data. It was found that GBT does allow a general view of the pixel data to be abstracted, that stray pixels being set and minor variations in line width are unimportant to the algorithms. The scan media for the test data sets were not edited in any way before scanning. Simple thresholding of the scanner output got rid of much of the extraneous data, for example, the grid lines on the circuit diagram. The rest was handled by the RVC software.

It was found that smoothing is a straightforward process using GBT arithmetic. It straightens lines and sharpens corners as might be expected. This smoothing process could be integrated into the line and boundary following algorithms to increase efficiency rather than performing it as a separate step.

Boundary-following is apparently easier than line-following. It was difficult to obtain samples in which boundary-following was required. The blob in Figure 50 was scanned from an ink drawing. Figure 51 shows its vectorization by the boundary-following algorithm while Figure 52 is a close-up of some of the boundary centroids.

Pattern types take on new importance in working with real data, while orientations are less important. Orientations are sensitive to stray pixels being set and to the pattern falling oddly in an aggregate. In the case of transit, pathological, and almost empty (whatever their classification) cells, the orientation is practically useless. On the other hand, by playing with the pattern type threshold function, pattern types can very effectively generalize or average the pixel data. Their use has the added advantage of reducing many operations to a table look-up. Nearly empty cells cause more trouble than expected. Ignoring them increases the number of line breaks and the inefficiencies in line-following. Giving them too much weight in the conversion process results in small triangles (see the corners of the blocks in Figure 40).

The idea of switching aggregate levels in processing lines of different width is important to the GBT approach. The actual mechanism for doing this became apparent in developing the existing algorithms. A pathological pattern type indicates the need to descend a level and process the seven subordinates rather than the parent. A full cell requires ascending a level to examine the parent cell. Since pattern types are in fact bit maps that tell which descendants are set, they are the key to this movement between levels. The level change logic was not implemented because it requires a complete restructuring of the code. This accounts for the problems evident in Figures 38 and 42. In Figure 38 the building symbols are poorly vectorized because the aggregates in those areas are full. In Figure 42 the letters run together because the pathologies were not handled properly. Most of these errors will be corrected when the level change algorithm is implemented.

Some problems are inevitable, especially without extensive pre-editing of the scan medium. There are also changes that should be made between hard copy and digital form. For example, the buildings in Figure 29 should not be present in the file as rectangles but should instead be represented as a location to which certain attributes are attached. The task of the vectorization algorithms then becomes one of transforming those pixels which make up a rectangle on the map into a form that is recognizable (by a pattern recognition algorithm) as a building symbol. The GBT data structure offers much power in designing algorithms to do these automatic editing functions. A GBT data base represents data as a set of points or locations. Each point knows how many and which points it is connected to. The points can be examined in some small region or generally, over a large area. These properties make it easy to do such things as find line endings which are within some distance of each other or find all triangles smaller than a certain size. The real strength of an automated entry system built around GBT will lie in intelligent and versatile editing algorithms.

To summarize, the results of this research support the idea that the GBT approach to raster to vector conversion is a good one. The algorithms handled a variety of data types well. Ideas for improving their performance evolved as more data was processed. It is planned that these improvements will be implemented. The results are strong enough to warrant further work toward a prototype RVC system.

6.0 Future Research Directions

In the course of this project, new areas of research were defined. Generally, they fall under the heading of the extension of the ideas and algorithms developed here to nonbinary images.

With the Optronics scanner, a gray scale value between 0 and 255 is measured for each pixel. The current procedure is to threshold this value so that the experimental system works with a binary picture function. With some minor modification the RVC software could handle a nonbinary function and line and edge extraction could be tried on such things as overhead photography. Working in this new setting one can also consider some generalizations of the aggregate descriptors used in RVC. An aggregate descriptor that is related to textural regions in an image would be very useful. For example, define a sequence of descriptors in this way. If A is a level n aggregate and k is an

integer, $0 \leq k \leq n$, define $M_k(A) = \frac{|\sum_s g_s|}{\sum_s |g_s|}$ where s ranges over the 7^{n-k}

level k aggregates subordinate to A and g_s is the gradient of the aggregate s . The function M_k in some sense measures the variability among the gradients of the level k subordinates of an aggregate.

$M_k(A)$ is 0 if the nonzero gradients sum to 0. Figures 53 and 54 show examples of these extreme cases.

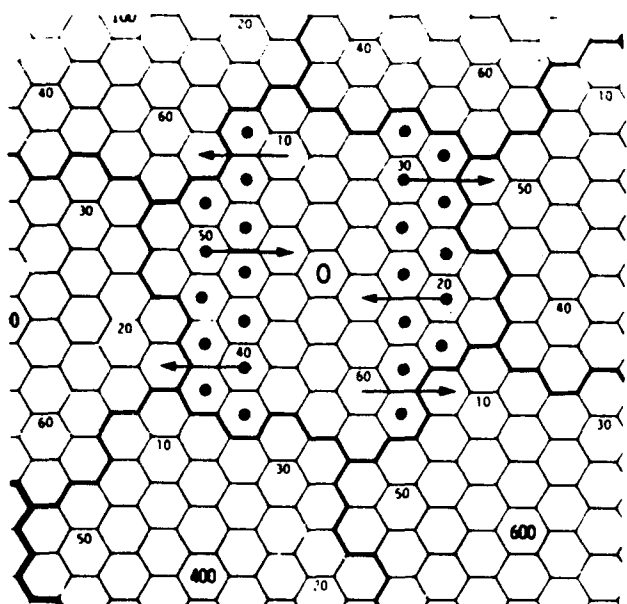


FIGURE 53: Aggregate A for which $M_1(A) = 0$

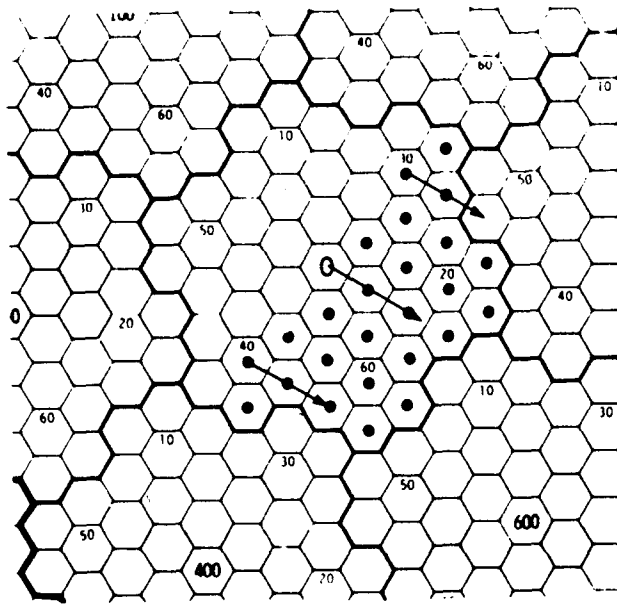


FIGURE 54: Aggregate A for which $M_1(A) = 1$

Using these functions M_k , identify with each level n aggregate, A , an $(n-1)$ -tuple, $M_1(A), \dots, M_{n-1}(A)$. If these tuples are regarded as points in $(n-1)$ -dimensional Euclidean space, they can be analyzed for clustering. Different clusters will tend to be characteristic of different textures.

Other functions that indicate gradient variability are

$G_k(A) = \sum s g_s$, the GBT vector sum of the level k gradients and

$T(A) = 1/7 |(7 \cdot g_0 - \sum_{i=1}^6 g_i)|$ where g_0, g_1, \dots, g_6 are the gradients of the seven immediate (level $n-1$) subordinates of A .

A second type of aggregate descriptor which can be defined is a function of weights rather than gradients. If A is a level n aggregate and B is the level $n+1$ aggregate which contains it, define

$V(A) = (w_A - \bar{w}_B)^2$ where w_A is the weight of A and \bar{w}_B is the average of the weights of the seven n th level subordinates of B .

Another function, VW , can be defined so that $VW(B) = \sum_s V(s)$ where s ranges over the seven subordinates. VW is then the variance of the weights.

The function V is used in the algorithms for binary images to define pattern types. It is possible to generalize this idea to the more complex patterns represented by various textures in an image. Thus, instead of 128 pattern types there will be thousands of texture types which can be defined using functions like the ones described above and combinations thereof.

6.1 Prototype Automated Graphics Entry System

Based on the results of this research, Interactive Systems Corporation is proceeding with the development of a prototype Automated Graphics Entry System. It will include an Optronics 40 inch by 48 inch optical scanner, a product which is itself in the final stages of development. The system will use either a DEC VAX 11/780 or a VAX 11/750. Peripherals such as large disks, a graphics workstation, and hard copy output devices will be incorporated.

Two properties essential to this prototype are speed and intelligence. Research indicates that the GBT approach can support a system that is both fast and has the self-editing and character recognition capability necessary to meet the needs of the user community. The speed requirement will be met by redesigning the general purpose data manager currently used. This DBMS was designed for a multiuser, multiple data base environment. It will be customized for the RVC application. Those processes that slow the raster to vector conversion will be identified. Firmware coding and the integration of microprocessors are possible means to speeding these "bottlenecks".

The extent of the intelligence possible with GBT is still being explored. GBT representations of line data facilitate the design of edit functions to remove line anomalies and clean up the vectorization. They are well suited to simple character and symbol recognition.

The result of the current efforts in design, algorithm development, and the integration of hardware, software, and firmware should be a system which is far more sophisticated than anything currently available.

References

1. Donna Peuquet, "An Examination of Techniques for Reformatting Digital Cartographic Data" to appear in Cartografica.
2. Dean Lucas, "Multiplication in N-Space", Proceedings of the American Mathematical Society, April 1979.
3. Dean Lucas and Laurie Gibson, "A System for Hierarchical Addressing in Euclidean Space", Interactive Systems Corporation, January 1980.
4. Laurie Gibson and Dean Lucas, "Vectorization of Raster Images Using Hierarchical Methods" to appear in Computer Graphics and Image Processing.